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)"), "${category}", 0),
75              // class GenericCollectionMethodsBase<TElement> {
76              // template <typename TElement> class GenericCollectionMethodsBase {
77              (new Regex(@"(?<before>\r?\n)(?<indent>[ \t]*)(?<type>class|struct)
                    (?<typeName>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9
                ↪  ,]+)>(?<typeDefinitionEnding>[^{]+){"), "${before}${indent}template <typename
                ↪  ...> ${type} ${typeName};" + Environment.NewLine + "${indent}template <typename
                ↪  ${typeParameters}> ${type}
                ↪  ${typeName}<${typeParameters}>${typeDefinitionEnding}{", 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($4)", 0),
81              // interface IFactory<out TProduct> {
82              // template <typename...> class IFactory;\ntemplate <typename TProduct> class
                ↪  IFactory<TProduct>
83              (new Regex(@"(?<before>\r?\n)(?<indent>[ \t]*)interface
                    (?<interface>[a-zA-Z0-9]+)<(?<typeParameters>[a-zA-Z0-9
                ↪  ,]+)>(?<typeDefinitionEnding>[^{]+){"), "${before}${indent}template <typename
                ↪  ...> class ${interface};" + Environment.NewLine + "${indent}template <typename
                ↪  ${typeParameters}> class
                ↪  ${interface}<${typeParameters}>${typeDefinitionEnding}{" + Environment.NewLine +
                ↪  "    public:", 0),
84              // template <typename TObject, TProperty, TValue>
85              // template <typename TObject, typename TProperty, typename TValue>
86              (new Regex(@"(?<before>template <(, )?typename [a-zA-Z0-9]+)+,
                ↪  )(?<typeParameter>[a-zA-Z0-9]+)(?<after>(,|>))"), "${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);
```

```
            // /*~extensionMethod~BuildExceptionString~*/...BuildExceptionString(sb,
            ↪  exception.InnerException, level + 1);
            (new Regex(@"(?<before>/\*~extensionMethod~(?<name>[a-zA-Z0-9]+)~\*/(.|\n)+\W)(?<var
            ↪  iable>[_a-zA-Z0-9]+)\.\k<name>\("), "${before}${name}(${variable}, ",
            ↪  50),
            // Remove markers
            // /*~extensionMethod~BuildExceptionString~*/
            //
            (new Regex(@"/\*~extensionMethod~[a-zA-Z0-9]+~\*/"), "", 0),
            // (this
            // (
            (new Regex(@"\(this "), "(", 0),
            // public: static readonly EnsureAlwaysExtensionRoot Always = new
            ↪  EnsureAlwaysExtensionRoot();
            // public: inline static EnsureAlwaysExtensionRoot Always;
            (new Regex(@"(?<access>(private|protected|public): )?static readonly
            ↪  (?<type>[a-zA-Z0-9]+(<[a-zA-Z0-9]+>)?) (?<name>[a-zA-Z0-9_]+) = new
            ↪  \k<type>\(\);"), "${access}inline static ${type} ${name};", 0),
            // public: static readonly Range<int> SByte = new
            ↪  Range<int>(std::numeric_limits<int>::min(), std::numeric_limits<int>::max());
            // public: inline static Range<int> SByte =
            ↪  Range<int>(std::numeric_limits<int>::min(), std::numeric_limits<int>::max());
            (new Regex(@"(?<access>(private|protected|public): )?static readonly
            ↪  (?<type>[a-zA-Z0-9]+(<[a-zA-Z0-9]+>)?) (?<name>[a-zA-Z0-9_]+) = new
            ↪  \k<type>\((?<arguments>[^\n]+)\);"), "${access}inline static ${type} ${name} =
            ↪  ${type}(${arguments});", 0),
            // public: static readonly string ExceptionContentsSeparator = "---";
            // public: inline static const char* ExceptionContentsSeparator = "---";
            (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),
            // private: const int MaxPath = 92;
            // private: inline static const int MaxPath = 92;
            (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),
            //  ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
            ↪  TArgument : class
            //  ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
            (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}*${after}",
            ↪  0),
            // protected: abstract TElement GetFirst();
            // protected: virtual TElement GetFirst() = 0;
            (new Regex(@"(?<access>(private|protected|public): )?abstract
            ↪  (?<method>[^;\r\n]+);"), "${access}virtual ${method} = 0;", 0),
            // TElement GetFirst();
            // virtual TElement GetFirst() = 0;
            (new Regex(@"([\r\n]+[ ]+)((?!return)[a-zA-Z0-9]+ [a-zA-Z0-9]+\([^\)\r\n]*\))(;[
            ↪  ]*[\r\n]+)"), "$1virtual $2 = 0$3", 1),
            // protected: readonly TreeElement[] _elements;
            // protected: TreeElement _elements[N];
            (new Regex(@"(?<access>(private|protected|public): )?readonly
            ↪  (?<type>[a-zA-Z<>0-9]+)([\[\]]+) (?<name>[_a-zA-Z0-9]+);"), "${access}${type}
            ↪  ${name}[N];", 0),
            // protected: readonly TElement Zero;
            // protected: TElement Zero;
            (new Regex(@"(?<access>(private|protected|public): )?readonly
            ↪  (?<type>[a-zA-Z<>0-9]+) (?<name>[_a-zA-Z0-9]+);"), "${access}${type} ${name};",
            ↪  0),
            // internal
            //
            (new Regex(@"(\W)internal\s+"), "$1", 0),
            // static void NotImplementedException(ThrowExtensionRoot root) => throw new
            ↪  NotImplementedException();
            // static void NotImplementedException(ThrowExtensionRoot root) { return throw new
            ↪  NotImplementedException(); }
            (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),
            // SizeBalancedTree(int capacity) => a = b;
            // SizeBalancedTree(int capacity) { a = b; }
            (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),
```

```
            // int SizeBalancedTree(int capacity) => a;
            // int SizeBalancedTree(int capacity) { return a; }
            (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),
            // () => Integer<TElement>.Zero,
            // () { return Integer<TElement>.Zero; },
            (new Regex(@"\(\)\s+=>\s+(?<expression>[^(),;\r\n]+(\(((?<parenthesis>\()|(?<-parenthesis>\))|[^();\r\n]*?)*?\))?[^(),;\r\n]*)(?<after>,|\);)"), "() { return ${expression}; }${after}", 0),
            // => Integer<TElement>.Zero;
            // { return Integer<TElement>.Zero; }
            (new Regex(@"\)\s+=>\s+([^;\r\n]+?);"), ") { return $1; }", 0),
            // () { return avlTree.Count; }
            // [&]()-> auto { return avlTree.Count; }
            (new Regex(@"(?<before>, |\()\(\) { return (?<expression>[^;\r\n]+); }"), "${before}[&]()-> auto { return ${expression}; }", 0),
            // Count => GetSizeOrZero(Root);
            // GetCount() { return GetSizeOrZero(Root); }
            (new Regex(@"(\W)([A-Z][a-zA-Z]+)\s+=>\s+([^;\r\n]+);"), "$1Get$2() { return $3; }", 0),
            // ArgumentInRange(const char* message) { const char* messageBuilder() { return message; }
            // ArgumentInRange(const char* message) { auto messageBuilder = [&]() -> const char* { return message; };
            (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]+""|[^}]|\n)+?)}"), "${before}auto ${methodName} = [&]() -> ${returnType} {${body}};", 10),
            // Func<TElement> treeCount
            // std::function<TElement()> treeCount
            (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", 0),
            // Action<TElement> free
            // std::function<void(TElement)> free
            (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<void($1)> $2", 0),
            // Predicate<TArgument> predicate
            // std::function<bool(TArgument)> predicate
            (new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<bool($1)> $2", 0),
            // var
            // auto
            (new Regex(@"(\W)var(\W)"), "$1auto$2", 0),
            // unchecked
            //
            (new Regex(@"[\r\n]{2}\s*?unchecked\s*?$"), "", 0),
            // throw new InvalidOperationException
            // throw std::runtime_error
            (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw std::runtime_error", 0),
            // void RaiseExceptionIgnoredEvent(Exception exception)
            // void RaiseExceptionIgnoredEvent(const std::exception& exception)
            (new Regex(@"(\(|, )(System\.Exception|Exception)( |\))"), "$1const std::exception&$3", 0),
            // EventHandler<Exception>
            // EventHandler<std::exception>
            (new Regex(@"(\W)(System\.Exception|Exception)(\W)"), "$1std::exception$3", 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 \*\+]+)(\([^\)\r\n]+?\))"), "$1$2 override", 0),
            // return (range.Minimum, range.Maximum)
            // return {range.Minimum, range.Maximum}
            (new Regex(@"(?<before>return\s*)\((?<values>[^\)\n]+)\)(?!\()(?<after>\W)"), "${before}{${values}}${after}", 0),
            // string
            // const char*
            (new Regex(@"(\W)string(\W)"), "$1const char*$2", 0),
            // System.ValueTuple
            // std::tuple
            (new Regex(@"(?<before>\W)(System\.)?ValueTuple(?!\s*=|\()(?<after>\W)"), "${before}std::tuple${after}", 0),
            // sbyte
            // std::int8_t
```

```
            (new Regex(@"(?<before>\W)((System\.)?SB|sb)yte(?!\s*=|\()(?<after>\W)"),
            ↪  "${before}std::int8_t${after}", 0),
            // short
            // std::int16_t
            (new Regex(@"(?<before>\W)((System\.)?Int16|short)(?!\s*=|\()(?<after>\W)"),
            ↪  "${before}std::int16_t${after}", 0),
            // int
            // std::int32_t
            (new Regex(@"(?<before>\W)((System\.)?I|i)nt(32)?(?!\s*=|\()(?<after>\W)"),
            ↪  "${before}std::int32_t${after}", 0),
            // long
            // std::int64_t
            (new Regex(@"(?<before>\W)((System\.)?Int64|long)(?!\s*=|\()(?<after>\W)"),
            ↪  "${before}std::int64_t${after}", 0),
            // byte
            // 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]+)"), "$1", 0),
            // std::int32_t.MinValue
            // std::numeric_limits<float>::min()
            (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+)\.MinValue(?<after>\W)"),
            ↪  "${before}std::numeric_limits<${type}>::min()${after}", 0),
            // float.MinValue
            // std::numeric_limits<float>::lowest()
            (new Regex(@"(?<before>\W)(?<type>float|double)\.MinValue(?<after>\W)"),
            ↪  "${before}std::numeric_limits<${type}>::lowest()${after}", 0),
            // double.MaxValue
            // std::numeric_limits<float>::max()
            (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MaxValue(?<after>\W
            ↪  )"), "${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(@"(struct|class) ([a-zA-Z0-9]+)(<[a-zA-Z0-9 ,]+>)? : ([a-zA-Z0-9]+)"),
            ↪  "$1 $2$3 : public $4", 0),
            // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
            // class IProperty : public ISetter<TValue, TObject>, public IProvider<TValue,
            ↪  TObject>
            (new Regex(@"(?<before>(struct|class) [a-zA-Z0-9]+ : ((public
            ↪  [a-zA-Z0-9]+(<[a-zA-Z0-9 ,]+>)?,
            ↪  )+)?)(?<inheritedType>(?!public)[a-zA-Z0-9]+(<[a-zA-Z0-9 ,]+>)?)(?<after>(,
            ↪  [a-zA-Z0-9]+(?!>)|[ \r\n]+))"), "${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
            //  TElement*
            (new Regex(@"( |\()ref ([a-zA-Z0-9]+) "), "$1$2* ", 0),
            // ref sizeBalancedTree.Root
            // &sizeBalancedTree->Root
            (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)"), "&$1->$2", 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", 0),
            // GetElement(node).Right
            // GetElement(node)->Right
            (new Regex(@"([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"), "$1($2)->$3", 0),
            // [Fact]\npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
            // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
            (new Regex(@"\[Fact\][\s\n]+(public: )?(static )?void ([a-zA-Z0-9]+)\(\)"), "public:
                TEST_METHOD($3)", 0),
            // class TreesTests
            // TEST_CLASS(TreesTests)
            (new Regex(@"class ([a-zA-Z0-9]+)Tests"), "TEST_CLASS($1)", 0),
            // Assert.Equal
            // Assert::AreEqual
            (new Regex(@"(Assert)\.Equal"), "$1::AreEqual", 0),
            // Assert.Throws
            // Assert::ExpectException
            (new Regex(@"(Assert)\.Throws"), "$1::ExpectException", 0),
            // $"Argument {argumentName} is null."
            // std::string("Argument
                ").append(Platform::Converters::To<std::string>(argumentName)).append(" is
                null.").data()
            (new Regex(@"\$""(?<left>(\\""|[^""\r\n])*){(?<expression>[_a-zA-Z0-9]+)}(?<right>(\
                \""|[^""\r\n])*)"""),
                "std::string($\"${left}\").append(Platform::Converters::To<std::string>(${expres
                sion})).append(\"${right}\").data()",
                10),
            // $"
            // "
            (new Regex(@"\$"""), "\"", 0),
            // std::string(std::string("[").append(Platform::Converters::To<std::string>(Minimum
                )).append(",
                ").data()).append(Platform::Converters::To<std::string>(Maximum)).append("]").da
                ta()
            // std::string("[").append(Platform::Converters::To<std::string>(Minimum)).append(",
                ").append(Platform::Converters::To<std::string>(Maximum)).append("]").data()
            (new Regex(@"std::string\((?<begin>std::string\(""(\\""|[^""])*""\)(\.append\((Platf
                orm::Converters::To<std::string>\([^)\n]+\)|[^)\n]+)\))+)\.data\(\)\)\.append"),
                "${begin}.append", 10),
            // Console.WriteLine("...")
            // printf("...\n")
            (new Regex(@"Console\.WriteLine\(""([^""\r\n]+)""\)"), "printf(\"$1\\n\")", 0),
            // TElement Root;
            // TElement Root = 0;
            (new Regex(@"(\r?\n[\t ]+)(private|protected|public)?(:
                )?([a-zA-Z0-9:_]+(?<!return)) ([_a-zA-Z0-9]+);"), "$1$2$3$4 $5 = 0;", 0),
            // TreeElement _elements[N];
            // TreeElement _elements[N] = { {0} };
```

```
309             (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),
310             // auto path = new TElement[MaxPath];
311             // TElement path[MaxPath] = { {0} };
312             (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),
313             // bool Equals(Range<T> other) { ... }
314             // bool operator ==(const Key &other) const { ... }
315             (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),
316             // Insert scope borders.
317             // class Range { ... public: override const char* ToString() { return ...; }
318             // class Range {/*~Range<T>~*/ ... public: override const char* ToString() { return
                ↪  ...; }
319             (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)template <typename
                    (?<typeParameter>[^<>\n]+)> (struct|class)
                ↪  (?<type>[a-zA-Z0-9]+<\k<typeParameter>>)(\s*:\s*[^{\n]+)?[\t ]*(\r?\n)?[\t
                ↪  ]*{)(?<middle>((?!class|struct).|\n)+?)(?<toStringDeclaration>(?<access>(private
                ↪  |protected|public): )override const char\* ToString\(\))"),
                ↪  "${classDeclarationBegin}/*~${type}~*/${middle}${toStringDeclaration}", 0),
320             // Inside the scope of ~!Range!~ replace:
321             // public: override const char* ToString() { return ...; }
322             // public: operator std::string() const { return ...; }\n\npublic: friend
                ↪  std::ostream & operator <<(std::ostream &out, const A &obj) { return out <<
                ↪  (std::string)obj; }
323             (new Regex(@"(?<scope>/\*~(?<type>[_a-zA-Z0-9<>:]+)~\*/)(?<separator>.|\n)(?<before>
                    ((?<!/\*~\k<type>~\*/)(.|\n))*?)(?<toStringDeclaration>\r?\n(?<indent>[
                ↪  \t]*)(?<access>(private|protected|public): )override const char\* ToString\(\)
                ↪  (?<toStringMethodBody>{[^}\n]+}))"), "${scope}${separator}${before}" +
                ↪  Environment.NewLine + "${indent}${access}operator std::string() const
                ↪  ${toStringMethodBody}" + Environment.NewLine + Environment.NewLine +
                ↪  "${indent}${access}friend std::ostream & operator <<(std::ostream &out, const
                ↪  ${type} &obj) { return out << (std::string)obj; }", 0),
324             // Remove scope borders.
325             // /*~Range~*/
326             //
327             (new Regex(@"/\*~[_a-zA-Z0-9<>:]+~\*/"), "", 0),
328             // private: inline static ConcurrentBag<std::exception> _exceptionsBag;
329             // private: inline static std::mutex _exceptionsBag_mutex; \n\n private: inline
                ↪  static std::vector<std::exception> _exceptionsBag;
330             (new Regex(@"(?<begin>\r?\n?(?<indent>[ \t]+))(?<access>(private|protected|public):
                    )?inline static ConcurrentBag<(?<argumentType>[^;\r\n]+)>
                ↪  (?<name>[_a-zA-Z0-9]+);"), "${begin}private: inline static std::mutex
                ↪  ${name}_mutex;" + Environment.NewLine + Environment.NewLine +
                ↪  "${indent}${access}inline static std::vector<${argumentType}> ${name};", 0),
331             // public: static IReadOnlyCollection<std::exception> GetCollectedExceptions() {
                ↪  return _exceptionsBag; }
332             // public: static std::vector<std::exception> GetCollectedExceptions() { return
                ↪  std::vector<std::exception>(_exceptionsBag); }
333             (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}>(${fieldName}); }", 0),
334             // public: static event EventHandler<std::exception> ExceptionIgnored =
                ↪  OnExceptionIgnored; ... };
335             // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
                ↪  const std::exception&)> ExceptionIgnored = OnExceptionIgnored; };
336             (new Regex(@"(?<begin>\r?\n(\r?\n)?(?<halfIndent>[
                    \t]+)\k<halfIndent>)(?<access>(private|protected|public): )?static event
                ↪  EventHandler<(?<argumentType>[^;\r\n]+)> (?<name>[_a-zA-Z0-9]+) = (?<defaultDele
                ↪  gate>[_a-zA-Z0-9]+);(?<middle>(.|\n)+?)(?<end>\r?\n\k<halfIndent>};)"),
                ↪  "${middle}" + Environment.NewLine + Environment.NewLine +
                ↪  "${halfIndent}${halfIndent}${access}static inline
                ↪  Platform::Delegates::MulticastDelegate<void(void*, const ${argumentType}&)>
                ↪  ${name} = ${defaultDelegate};${end}", 0),
337             // Insert scope borders.
338             // class IgnoredExceptions { ... private: inline static std::vector<std::exception>
                ↪  _exceptionsBag;
339             // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: inline static
                ↪  std::vector<std::exception> _exceptionsBag;
```

```
340             (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),
341             // Inside the scope of ~!_exceptionsBag!~ replace:
342             // _exceptionsBag.Add(exception);
343             // _exceptionsBag.push_back(exception);
344             (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),
345             // Remove scope borders.
346             // /*~_exceptionsBag~*/
347             //
348             (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
349             // Insert scope borders.
350             // class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
351             // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: static std::mutex
                    _exceptionsBag_mutex;
352             (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),
353             // Inside the scope of ~!_exceptionsBag!~ replace:
354             // return std::vector<std::exception>(_exceptionsBag);
355             // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
                    std::vector<std::exception>(_exceptionsBag);
356             (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);${after}", 10),
357             // Inside the scope of ~!_exceptionsBag!~ replace:
358             // _exceptionsBag.Add(exception);
359             // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
                    _exceptionsBag.Add(exception);
360             (new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<befor
                    e>((?<!/\*~\k<fieldName>~\*/)(.|\n))*?){(?<after>((?!lock_guard)([^{};]|\n))*?\r
                    ?\n(?<indent>[ \t]*)\k<fieldName>[^;}\r\n]*;)"),
                    "${scope}${separator}${before}{" + Environment.NewLine +
                    "${indent}std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", 10),
361             // Remove scope borders.
362             // /*~_exceptionsBag~*/
363             //
364             (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
365             // Insert scope borders.
366             // class IgnoredExceptions { ... public: static inline
                    Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
                    ExceptionIgnored = OnExceptionIgnored;
367             // class IgnoredExceptions {/*~ExceptionIgnored~*/ ... public: static inline
                    Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
                    ExceptionIgnored = OnExceptionIgnored;
368             (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n]+\r\n[\t
                    ]*{)(?<middle>((?!class).|\n)+?)(?<eventDeclaration>(?<access>(private|protected
                    |public): )static inline
                    Platform::Delegates::MulticastDelegate<(?<argumentType>[^;\r\n]+)>
                    (?<name>[_a-zA-Z0-9]+) = (?<defaultDelegate>[_a-zA-Z0-9]+);)"),
                    "${classDeclarationBegin}/*~${name}~*/${middle}${eventDeclaration}", 0),
369             // Inside the scope of ~!ExceptionIgnored!~ replace:
370             // ExceptionIgnored.Invoke(NULL, exception);
371             // ExceptionIgnored(NULL, exception);
372             (new Regex(@"(?<scope>/\*~(?<eventName>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before
                    >((?<!/\*~\k<eventName>~\*/)(.|\n))*?)\k<eventName>\.Invoke"),
                    "${scope}${separator}${before}${eventName}", 10),
373             // Remove scope borders.
374             // /*~ExceptionIgnored~*/
375             //
376             (new Regex(@"/\*~[a-zA-Z0-9]+~\*/"), "", 0),
377             // Insert scope borders.
378             // auto added = new StringBuilder();
379             // /*~sb~*/std::string added;
380             (new Regex(@"(auto|(System\.Text\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new
                    (System\.Text\.)?StringBuilder\(\);"), "/*~${variable}~*/std::string
                    ${variable};", 0),
381             // static void Indent(StringBuilder sb, int level)
382             // static void Indent(/*~sb~*/StringBuilder sb, int level)
```

```
            (new Regex(@"(?<start>, |\()(System\.Text\.)?StringBuilder
            ↪  (?<variable>[a-zA-Z0-9]+)(?<end>,|\))"), "${start}/*~${variable}~*/std::string&
            ↪  ${variable}${end}", 0),
            // Inside the scope of ~!added!~ replace:
            // sb.ToString()
            // sb.data()
            (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>
            ↪  ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.ToString\(\)"),
            ↪  "${scope}${separator}${before}${variable}.data()", 10),
            // sb.AppendLine(argument)
            // sb.append(Platform::Converters::To<std::string>(argument)).append(1, '\n')
            (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(Platform::Converters::To<std::s
            ↪  tring>(${argument})).append(1, '\\n')",
            ↪  10),
            // sb.Append('\t', level);
            // sb.append(level, '\t');
            (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),
            // sb.Append(argument)
            // sb.append(Platform::Converters::To<std::string>(argument))
            (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(Platform::Converters::To<std::s
            ↪  tring>(${argument}))",
            ↪  10),
            // Remove scope borders.
            // /*~sb~*/
            //
            (new Regex(@"/\*~[a-zA-Z0-9]+~\*/"), "", 0),
            // Insert scope borders.
            // auto added = new HashSet<TElement>();
            // ~!added!~std::unordered_set<TElement> added;
            (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
            ↪  HashSet<(?<element>[a-zA-Z0-9]+)>\(\);"),
            ↪  "~!${variable}!~std::unordered_set<${element}> ${variable};", 0),
            // Inside the scope of ~!added!~ replace:
            // added.Add(node)
            // added.insert(node)
            (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),
            // Inside the scope of ~!added!~ replace:
            // added.Remove(node)
            // added.erase(node)
            (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),
            // 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});", 0),
            // Remove scope borders.
            // ~!added!~
            //
            (new Regex(@"~![a-zA-Z0-9]+!~"), "", 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);", 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))*?)(Platform::Converters::To<std::string>\(\k<
                variable>\.Message\)|\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),
            // throw new ArgumentOutOfRangeException(argumentName, argumentValue,
                messageBuilder());
            // throw std::invalid_argument(std::string("Value
                [").append(Platform::Converters::To<std::string>(argumentValue)).append("] of
                argument [").append(argumentName).append("] is out of range:
                ").append(messageBuilder()).append("."));
```

```
469             (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(Platform::Converters::To<std::string>(${argumentValue})).append(\"]
                ↪  of argument [\").append(${argument}).append(\"] is out of range:
                ↪  \").append(${message}).append(\".\"));", 0),
470             // throw new NotSupportedException();
471             // throw std::logic_error("Not supported exception.");
472             (new Regex(@"throw new NotSupportedException\(\);"), "throw std::logic_error(\"Not
                ↪  supported exception.\");", 0),
473             // throw new NotImplementedException();
474             // throw std::logic_error("Not implemented exception.");
475             (new Regex(@"throw new NotImplementedException\(\);"), "throw std::logic_error(\"Not
                ↪  implemented exception.\");", 0),
476             // Insert scope borders.
477             // const std::string& message
478             // const std::string& message/*~message~*/
479             (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?((std::)?string&?|char\*)
                ↪  (?<variable>[_a-zA-Z0-9]+))(?<after>\W)"),
                ↪  "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
480             // Inside the scope of /*~message~*/ replace:
481             // Platform::Converters::To<std::string>(message)
482             // message
483             (new Regex(@"(?<scope>/\*~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before
                    >((?<!/\*~\k<variable>~\*/)(.|\n))*?)Platform::Converters::To<std::string>\(\k<v
                ↪  ariable>\)"), "${scope}${separator}${before}${variable}",
                ↪  10),
484             // Remove scope borders.
485             // /*~ex~*/
486             //
487             (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
488             // Insert scope borders.
489             // std::tuple<T, T> tuple
490             // std::tuple<T, T> tuple/*~tuple~*/
491             (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?tuple<[^\n]+>&?
                ↪  (?<variable>[_a-zA-Z0-9]+))(?<after>\W)"),
                ↪  "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
492             // Inside the scope of ~!ex!~ replace:
493             // tuple.Item1
494             // std::get<1-1>(tuple)
495             (new Regex(@"(?<scope>/\*~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before
                    >((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.Item(?<itemNumber>\d+)(?<afte
                ↪  r>\W)"),
                ↪  "${scope}${separator}${before}std::get<${itemNumber}-1>(${variable})${after}",
                ↪  10),
496             // Remove scope borders.
497             // /*~ex~*/
498             //
499             (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
500             // Insert scope borders.
501             // class Range<T> {
502             // class Range<T> {/*~type~Range<T>~*/
503             (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)template <typename
                ↪  (?<typeParameter>[^\n]+)> (struct|class)
                ↪  (?<type>[a-zA-Z0-9]+<\k<typeParameter>>)(\s*:\s*[^{\n]+)?[\t ]*(\r?\n)?[\t
                ↪  ]*{)"), "${classDeclarationBegin}/*~type~${type}~*/", 0),
504             // Inside the scope of /*~type~Range<T>~*/ insert inner scope and replace:
505             // public: static implicit operator std::tuple<T, T>(Range<T> range)
506             // public: operator std::tuple<T, T>() const {/*~variable~Range<T>~*/
507             (new Regex(@"(?<scope>/\*~type~(?<type>[^~\n\*]+)~\*/)(?<separator>.|\n)(?<before>((
                    ?<!/\*~type~\k<type>~\*/)(.|\n))*?)(?<access>(private|protected|public): )static
                ↪  implicit operator (?<targetType>[^\(\n]+)\((?<argumentDeclaration>\k<type>
                ↪  (?<variable>[a-zA-Z0-9]+))\)(?<after>\s*\n?\s*{)"),
                ↪  "${scope}${separator}${before}${access}operator ${targetType}()
                ↪  const${after}/*~variable~${variable}~*/", 10),
508             // Inside the scope of /*~type~Range<T>~*/ replace:
509             // public: static implicit operator Range<T>(std::tuple<T, T> tuple) { return new
                ↪  Range<T>(std::get<1-1>(tuple), std::get<2-1>(tuple)); }
510             // public: Range(std::tuple<T, T> tuple) : Range(std::get<1-1>(tuple),
                ↪  std::get<2-1>(tuple)) { }
```

```
511             (new Regex(@"(?<scope>/\*~type~(?<type>(?<typeName>[_a-zA-Z0-9]+)[^~\n\*]*)~\*/)(?<separator>.|\n)(?<before>((?<!/\*~type~\k<type>~\*/)(.|\n))*?)(?<access>(private|protected|public): )static implicit operator \k<type>\((?<arguments>[^{}\n]+)\)(\s|\n)*{(\s|\n)*return (new )?\k<type>\((?<passedArguments>[^\n]+)\);(\s|\n)*}"),
                "${scope}${separator}${before}${access}${typeName}(${arguments}) : ${typeName}(${passedArguments}) { }", 10),
512             // Inside the scope of /*~variable~range~*/ replace:
513             // range.Minimum
514             // this->Minimum
515             (new Regex(@"(?<scope>{/\*~variable~(?<variable>[^~\n]+)~\*/)(?<separator>.|\n)(?<before>(?<beforeExpression>(?<bracket>{)|(?<-bracket>})|[^{}]|\n)*?)\k<variable>\.(?<field>[_a-zA-Z0-9]+)(?<after>(,|;|}|
                |\))(?<afterExpression>(?<bracket>{)|(?<-bracket>})|[^{}]|\n)*?})"),
                "${scope}${separator}${before}this->${field}${after}", 10),
516             // Remove scope borders.
517             // /*~ex~*/
518             //
519             (new Regex(@"/\*~[^~\n]+~[^~\n]+~\*/"), "", 0),
520         }.Cast<ISubstitutionRule>().ToList();
521
522         public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
523         {
524             // ICounter<int, int> c1;
525             // ICounter<int, int>* c1;
526             (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^>\r\n]+>)?)
                (?<variable>[_a-zA-Z0-9]+)(?<after> = null)?;"), "${abstractType}*
                ${variable}${after};", 0),
527             // (expression)
528             // expression
529             (new Regex(@"(\(| )\(([a-zA-Z0-9_\*:]+)\)(,| |;|\))"), "$1$2$3", 0),
530             // (method(expression))
531             // method(expression)
532             (new Regex(@"(?<firstSeparator>(\(|
                ))\((?<method>[a-zA-Z0-9_\->\*:]+)\((?<expression>((?<parenthesis>\()|(?<-parenthesis>\))|[a-zA-Z0-9_\->\*:]*)+)(?(parenthesis)(?!))\)\)(?<lastSeparator>(,|
                |;|\)))"), "${firstSeparator}${method}(${expression})${lastSeparator}", 0),
533             // .append(".")
534             // .append(1, '.');
535             (new Regex(@"\.append\(""([^\\""]|\\[^""])""\)"), ".append(1, '$1')", 0),
536             // return ref _elements[node];
537             // return &_elements[node];
538             (new Regex(@"return ref ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9\*]+)\];"), "return &$1[$2];",
                0),
539             // null
540             // nullptr
541             (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)null
                (?<after>\W)"), "${before}nullptr${after}",
                10),
542             // default
543             // 0
544             (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)defa
                ult(?<after>\W)"), "${before}0${after}",
                10),
545             // object x
546             // void *x
547             (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)([O|
                o]bject|System\.Object) (?<after>\w)"), "${before}void *${after}",
                10),
548             // <object>
549             // <void*>
550             (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(?<!
                \w )([O|o]bject|System\.Object)(?<after>\W)"), "${before}void*${after}",
                10),
551             // ArgumentNullException
552             // std::invalid_argument
553             (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(Sys
                tem\.)?ArgumentNullException(?<after>\W)"),
                "${before}std::invalid_argument${after}", 10),
554             // template <typename T> struct Range : IEquatable<Range<T>>
555             // template <typename T> struct Range {
556             (new Regex(@"(?<before>template <typename (?<typeParameter>[^\n]+)> (struct|class)
                (?<type>[a-zA-Z0-9]+<[^\n]+>)) : (public
                )?IEquatable<\k<type>>(?<after>(\s|\n)*{)"), "${before}${after}", 0),
557             // #region Always
558             //
```

```
            (new Regex(@"(^|\r?\n)[ \t]*\#(region|endregion)[^\r\n]*(\r?\n|$)"), "", 0),
            // //#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"), "$1class", 0),
            // \n\n\n
            // \n\n
            (new Regex(@"\r?\n[ \t]*\r?\n[ \t]*\r?\n"), Environment.NewLine + Environment.NewLine, 50),
            // {\n\n
            // {\n
            (new Regex(@"{[ \t]*\r?\n[ \t]*\r?\n"), "{" + Environment.NewLine, 10),
            // \n\n}
            // {\n
            (new Regex(@"\r?\n[ \t]*\r?\n(?<end>[ \t]*})"), Environment.NewLine + "${end}", 10),
        }.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