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]*((?<=\s)|\W)(interface|class|struct)(\W)[^\{\(\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);
```

```
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             // private: static readonly Disposal _emptyDelegate = (manual, wasDisposed) => { };
104             // private: inline static std::function<Disposal> _emptyDelegate = [](auto manual,
                ↪  auto wasDisposed) { };
105             (new Regex(@"(?<access>(private|protected|public): )?static readonly
                    (?<type>[a-zA-Z][a-zA-Z0-9]*) (?<name>[a-zA-Z_][a-zA-Z0-9_]*) =
                ↪  \((?<firstArgument>[a-zA-Z_][a-zA-Z0-9_]*),
                ↪  (?<secondArgument>[a-zA-Z_][a-zA-Z0-9_]*)\) => {\s*};"), "${access}inline static
                ↪  std::function<${type}> ${name} = [](auto ${firstArgument}, auto
                ↪  ${secondArgument}) { };", 0),
106             // public: static readonly EnsureAlwaysExtensionRoot Always = new
                ↪  EnsureAlwaysExtensionRoot();
107             // public: inline static EnsureAlwaysExtensionRoot Always;
108             (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),
109             // public: static readonly Range<int> SByte = new
                ↪  Range<int>(std::numeric_limits<int>::min(), std::numeric_limits<int>::max());
110             // public: inline static Range<int> SByte =
                ↪  Range<int>(std::numeric_limits<int>::min(), std::numeric_limits<int>::max());
111             (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),
112             // public: static readonly string ExceptionContentsSeparator = "---";
113             // public: inline static std::string ExceptionContentsSeparator = "---";
114             (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly) string
                ↪  (?<name>[a-zA-Z0-9_]+) = ""(?<string>(\\""|[^""\r\n])+)"";"), "${access}inline
                ↪  static std::string ${name} = \"${string}\";", 0),
115             // private: const int MaxPath = 92;
116             // private: inline static const int MaxPath = 92;
117             (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),
118             //  ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
                ↪  TArgument : class
119             //  ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
120             (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}*${after}",
                ↪  0),
121             // protected: abstract TElement GetFirst();
122             // protected: virtual TElement GetFirst() = 0;
123             (new Regex(@"(?<access>(private|protected|public): )?abstract
                ↪  (?<method>[^;\r\n]+);"), "${access}virtual ${method} = 0;", 0),
124             // TElement GetFirst();
125             // virtual TElement GetFirst() = 0;
126             (new Regex(@"(?<before>[\r\n]+[ ]+)(?<methodDeclaration>(?!return)[a-zA-Z0-9]+
                ↪  [a-zA-Z0-9]+\([^\)\r\n]*\))(?<after>;[ ]*[\r\n]+)"), "${before}virtual
                ↪  ${methodDeclaration} = 0${after}", 1),
127             // protected: readonly TreeElement[] _elements;
128             // protected: TreeElement _elements[N];
129             (new Regex(@"(?<access>(private|protected|public): )?readonly
                ↪  (?<type>[a-zA-Z<>0-9]+)([\[\]]+) (?<name>[_a-zA-Z0-9]+);"), "${access}${type}
                ↪  ${name}[N];", 0),
130             // protected: readonly TElement Zero;
131             // protected: TElement Zero;
132             (new Regex(@"(?<access>(private|protected|public): )?readonly
                ↪  (?<type>[a-zA-Z<>0-9]+) (?<name>[_a-zA-Z0-9]+);"), "${access}${type} ${name};",
                ↪  0),
133             // internal
134             //
135             (new Regex(@"(\W)internal\s+"), "$1", 0),
136             // static void NotImplementedException(ThrowExtensionRoot root) => throw new
                ↪  NotImplementedException();
```

```
137             // static void NotImplementedException(ThrowExtensionRoot root) { return throw new
                ↪  NotImplementedException(); }
138             (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),
139             // SizeBalancedTree(int capacity) => a = b;
140             // SizeBalancedTree(int capacity) { a = b; }
141             (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),
142             // int SizeBalancedTree(int capacity) => a;
143             // int SizeBalancedTree(int capacity) { return a; }
144             (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),
145             // OnDispose = (manual, wasDisposed) =>
146             // OnDispose = [&](auto manual, auto wasDisposed)
147             (new Regex(@"(?<variable>[a-zA-Z_][a-zA-Z0-9_]*)(?<operator>\s*\+?=\s*)\((?<firstArg
                ↪  ument>[a-zA-Z_][a-zA-Z0-9_]*),
                ↪  (?<secondArgument>[a-zA-Z_][a-zA-Z0-9_]*)\)\s*=>"),
                ↪  "${variable}${operator}[&](auto ${firstArgument}, auto ${secondArgument})", 0),
148             // () => Integer<TElement>.Zero,
149             // () { return Integer<TElement>.Zero; },
150             (new Regex(@"\(\)\s+=>\s+(?<expression>[^(),;\r\n]+(\(((?<parenthesis>\()|(?<-parent
                ↪  hesis>\))|[^();\r\n]*?)*?\))?[^(),;\r\n]*)(?<after>,|\);)"), "() { return
                ↪  ${expression}; }${after}", 0),
151             // ~DisposableBase() => Destruct();
152             // ~DisposableBase() { Destruct(); }
153             (new Regex(@"~(?<class>[a-zA-Z_][a-zA-Z0-9_]*)\(\)\s+=>\s+([^;\r\n]+?);"),
                ↪  "~${class}() { $1; }", 0),
154             // => Integer<TElement>.Zero;
155             // { return Integer<TElement>.Zero; }
156             (new Regex(@"\)\s+=>\s+([^;\r\n]+?);"), ") { return $1; }", 0),
157             // () { return avlTree.Count; }
158             // [&]()-> auto { return avlTree.Count; }
159             (new Regex(@"(?<before>, |\()\(\) { return (?<expression>[^;\r\n]+); }"),
                ↪  "${before}[&]()-> auto { return ${expression}; }", 0),
160             // Count => GetSizeOrZero(Root);
161             // Count() { return GetSizeOrZero(Root); }
162             (new Regex(@"(\W)([A-Z][a-zA-Z]+)\s+=>\s+([^;\r\n]+);"), "$1$2() { return $3; }", 0),
163             // public: T Object { get; }
164             // public: const T Object;
165             (new Regex(@"(?<before>[^\r]\r?\n[ \t]*)(?<access>(private|protected|public):
                ↪  )?(?<type>[a-zA-Z_][a-zA-Z0-9_:<>]*)
                ↪  (?<property>[a-zA-Z_][a-zA-Z0-9_]*)(?<blockOpen>[\n\s]*{[\n\s]*)(\[[^\n]+\][\n\s
                ↪  ]*)?get;(?<blockClose>[\n\s]*}(?<after>[\n\s]*)"), "${before}${access}const
                ↪  ${type} ${property};${after}", 2),
166             // public: bool IsDisposed { get => _disposed > 0; }
167             // public: bool IsDisposed() { return _disposed > 0; }
168             (new Regex(@"(?<before>[^\r]\r?\n[ \t]*)(?<access>(private|protected|public):
                ↪  )?(?<virtual>virtual )?bool
                ↪  (?<property>[a-zA-Z_][a-zA-Z0-9_]*)(?<blockOpen>[\n\s]*{[\n\s]*)(\[[^\n]+\][\n\s
                ↪  ]*)?get\s*=>\s*(?<expression>[^\n]+);(?<blockClose>[\n\s]*}[\n\s]*)"),
                ↪  "${before}${access}${virtual}bool ${property}()${blockOpen}return
                ↪  ${expression};${blockClose}", 2),
169             // protected: virtual std::string ObjectName { get => GetType().Name; }
170             // protected: virtual std::string ObjectName() { return GetType().Name; }
171             (new Regex(@"(?<before>[^\r]\r?\n[ \t]*)(?<access>(private|protected|public):
                ↪  )?(?<virtual>virtual )?(?<type>[a-zA-Z_][a-zA-Z0-9_:<>]*)
                ↪  (?<property>[a-zA-Z_][a-zA-Z0-9_]*)(?<blockOpen>[\n\s]*{[\n\s]*)(\[[^\n]+\][\n\s
                ↪  ]*)?get\s*=>\s*(?<expression>[^\n]+);(?<blockClose>[\n\s]*}[\n\s]*)"),
                ↪  "${before}${access}${virtual}${type} ${property}()${blockOpen}return
                ↪  ${expression};${blockClose}", 2),
172             // ArgumentInRange(string message) { string messageBuilder() { return message; }
173             // ArgumentInRange(string message) { auto messageBuilder = [&]() -> string { return
                ↪  message; };
174             (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),
175             // Func<TElement> treeCount
176             // std::function<TElement()> treeCount
177             (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(<(?<typeParameters>[a-zA-Z0-9]+(,
                ↪  ([a-zA-Z0-9]+))*)>)?(?<after>>| (?<variable>[a-zA-Z0-9]+))"),
                ↪  "std::function<void(${typeParameters})>${after}", 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
                // throw
                (new Regex(@"(\W)throw new(\W)"), "$1throw$2", 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}}${after}", 0),
                // string
                // std::string
                (new Regex(@"(?<before>\W)(?<!::)string(?<after>\W)"),
                ↪  "${before}std::string${after}", 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),
                // float.MinValue
```

```
239             // std::numeric_limits<float>::lowest()
240             (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MinValue(?<after>\W)"), "${before}std::numeric_limits<${type}>::lowest()${after}",
241             0),
242             // double.MaxValue
243             // std::numeric_limits<float>::max()
244             (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MaxValue(?<after>\W)"), "${before}std::numeric_limits<${type}>::max()${after}",
245             0),
246             // using Platform.Numbers;
247             //
248             (new Regex(@"([\r\n]{2}|^)\s*?using [\.a-zA-Z0-9]+;\s*?$"), "", 0),
249             // struct TreeElement { }
250             // struct TreeElement { };
251             (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([^;])"), "$1 $2$3{$4};$5", 0),
252             // class Program { }
253             // class Program { };
254             (new Regex(@"(?<type>struct|class)
255                 (?<name>[a-zA-Z0-9]+[^\r\n]*)(?<beforeBody>[\r\n]+(?<indentLevel>[\t]*)?)\{(?<body>[\S\s]+?[\r\n]+\k<indentLevel>)\}(?<afterBody>[^;]|$)"), "${type} ${name}${beforeBody}{${body}};${afterBody}", 0),
256             // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
257             // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
258             (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(<[a-zA-Z0-9 ,]+>)? : ([a-zA-Z0-9]+)"), "$1 $2$3 : public $4", 0),
259             // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
260             // class IProperty : public ISetter<TValue, TObject>, public IProvider<TValue, TObject>
261             (new Regex(@"(?<before>(struct|class) [a-zA-Z0-9]+ : ((public [a-zA-Z0-9]+(<[a-zA-Z0-9 ,]+>)?, )+)?)(?<inheritedType>(?!public)[a-zA-Z0-9]+(<[a-zA-Z0-9 ,]+>)?)(?<after>(, [a-zA-Z0-9]+(?!>)|[ \r\n]+))"), "${before}public ${inheritedType}${after}", 10),
262             // Insert scope borders.
263             // ref TElement root
264             // ~!root!~ref TElement root
265             (new Regex(@"(?<definition>(?<= |\()(ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!ref)) (?<variable>[a-zA-Z0-9]+)(?=\)|, | =))"), "~!${variable}!~${definition}", 0),
266             // Inside the scope of ~!root!~ replace:
267             // root
268             // *root
269             (new Regex(@"(?<definition>~!(?<pointer>[a-zA-Z0-9]+)!~ref [a-zA-Z0-9]+ \k<pointer>(?=\)|, | =))(?<before>((?<!~!\k<pointer>!~)(.|\n))*?)(?<prefix>(\W|\())\k<pointer>(?<suffix>( |\)|;|,))"),
270             "${definition}${before}${prefix}*${pointer}${suffix}", 70),
271             // Remove scope borders.
272             // ~!root!~
273             //
274             (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", 5),
275             // ref auto root = ref
276             // ref auto root =
277             (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)"), "$1* $2 =$3", 0),
278             // *root = ref left;
279             // root = left;
280             (new Regex(@"\*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", 0),
281             // (ref left)
282             // (left)
283             (new Regex(@"\(ref ([a-zA-Z0-9]+)(\)|\(|,)"), "($1$2", 0),
284             //  ref TElement
285             //  TElement*
286             (new Regex(@"( |\()ref ([a-zA-Z0-9]+) "), "$1$2* ", 0),
287             // ref sizeBalancedTree.Root
288             // &sizeBalancedTree->Root
289             (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)"), "&$1->$2", 0),
290             // ref GetElement(node).Right
291             // &GetElement(node)->Right
292             (new Regex(@"ref ([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"), "&$1($2)->$3", 0),
293             // GetElement(node).Right
294             // GetElement(node)->Right
295             (new Regex(@"([a-zA-Z0-9]+)\(([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)"), "$1($2)->$3", 0),
296             // [Fact]\npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
297             // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
298             (new Regex(@"\[Fact\][\s\n]+(public: )?(static )?void ([a-zA-Z0-9]+)\(\)"), "public: TEST_METHOD($3)", 0),
299             // class TreesTests
```

```
            // TEST_CLASS(TreesTests)
            (new Regex(@"class ([a-zA-Z0-9]+Tests)"), "TEST_CLASS($1)", 0),
            // Assert.Equal
            // Assert::AreEqual
            (new Regex(@"(?<type>Assert)\.(?<method>(Not)?Equal)"), "${type}::Are${method}", 0),
            // Assert.Throws
            // Assert::ExpectException
            (new Regex(@"(Assert)\.Throws"), "$1::ExpectException", 0),
            // Assert.True
            // Assert::IsTrue
            (new Regex(@"(Assert)\.(True|False)"), "$1::Is$2", 0),
            // $"Argument {argumentName} is null."
            // std::string("Argument ").append(Platform::Converters::To<std::string>(argumentName)).append(" is null.")
            (new Regex(@"\$""(?<left>(\\""|[^""\r\n])*){(?<expression>[_a-zA-Z0-9]+)}(?<right>(\\""|[^""\r\n])*)"""),
            "std::string($\"${left}\").append(Platform::Converters::To<std::string>(${expression})).append(\"${right}\")",
            10),
            // $"
            // "
            (new Regex(@"\$"""), "\"", 0),
            // std::string(std::string("[").append(Platform::Converters::To<std::string>(Minimum)).append(",
            // ")).append(Platform::Converters::To<std::string>(Maximum)).append("]")
            // std::string("[").append(Platform::Converters::To<std::string>(Minimum)).append(",
            // ").append(Platform::Converters::To<std::string>(Maximum)).append("]")
            (new Regex(@"std::string\((?<begin>std::string\(""(\\""|[^""])*""\)(\.append\((Platform::Converters::To<std::string>\([^)\n]+\)|[^)\n]+)\))+)\)\.append"),
            "${begin}.append", 10),
            // Console.WriteLine("...")
            // printf("...\n")
            (new Regex(@"Console\.WriteLine\(""([^""\r\n]+)""\)"), "printf(\"$1\\n\")", 0),
            // TElement Root;
            // TElement Root = 0;
            (new Regex(@"(?<before>\r?\n[\t ]+)(?<access>(private|protected|public)(: )?)?(?<type>[a-zA-Z0-9:_]+(?<!return)) (?<name>[_a-zA-Z0-9]+);"),
            "${before}${access}${type} ${name} = 0;", 0),
            // TreeElement _elements[N];
            // TreeElement _elements[N] = { {0} };
            (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),
            // auto path = new TElement[MaxPath];
            // TElement path[MaxPath] = { {0} };
            (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),
            // bool Equals(Range<T> other) { ... }
            // bool operator ==(const Key &other) const { ... }
            (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),
            // Insert scope borders.
            // class Range { ... public: override std::string ToString() { return ...; }
            // class Range {/*~Range<T>~*/ ... public: override std::string ToString() { return ...; }
            (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 std::string ToString\(\))"),
            "${classDeclarationBegin}/*~${type}~*/${middle}${toStringDeclaration}", 0),
            // Inside the scope of ~!Range!~ replace:
            // public: override std::string ToString() { return ...; }
            // public: operator std::string() const { return ...; }\n\npublic: friend std::ostream & operator <<(std::ostream &out, const A &obj) { return out << (std::string)obj; }
            (new Regex(@"(?<scope>/\*~(?<type>[_a-zA-Z0-9<>:]+)~\*/)(?<separator>.|\n)(?<before>((?<!/\*~\k<type>~\*/)(.|\n))*?)(?<toStringDeclaration>\r?\n(?<indent>[\t]*)(?<access>(private|protected|public): )override std::string 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),
```

```
339             // Remove scope borders.
340             // /*~Range~*/
341             //
342             (new Regex(@"/\*~[_a-zA-Z0-9<>:]+~\*/"), "", 0),
343             // private: inline static ConcurrentBag<std::exception> _exceptionsBag;
344             // private: inline static std::mutex _exceptionsBag_mutex; \n\n private: inline
                ↪  static std::vector<std::exception> _exceptionsBag;
345             (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),
346             // public: static IReadOnlyCollection<std::exception> GetCollectedExceptions() {
                ↪  return _exceptionsBag; }
347             // public: static std::vector<std::exception> GetCollectedExceptions() { return
                ↪  std::vector<std::exception>(_exceptionsBag); }
348             (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}>(${fieldName}); }", 0),
349             // public: static event EventHandler<std::exception> ExceptionIgnored =
                ↪  OnExceptionIgnored; ... };
350             // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
                ↪  const std::exception&)> ExceptionIgnored = OnExceptionIgnored; };
351             (new Regex(@"(?<begin>\r?\n(\r?\n)?(?<halfIndent>[
                    \t]+)\k<halfIndent>)(?<access>(private|protected|public): )?static event
                ↪  EventHandler<(?<argumentType>[^;\r\n]+)> (?<name>[_a-zA-Z0-9]+) = (?<defaultDele
                ↪  gate>[_a-zA-Z0-9]+);(?<middle>(.|\n)+?)(?<end>\r?\n\k<halfIndent>};)"),
                ↪  "${middle}" + Environment.NewLine + Environment.NewLine +
                ↪  "${halfIndent}${halfIndent}${access}static inline
                ↪  Platform::Delegates::MulticastDelegate<void(void*, const ${argumentType}&)>
                ↪  ${name} = ${defaultDelegate};${end}", 0),
352             // public: event Disposal OnDispose;
353             // public: Platform::Delegates::MulticastDelegate<Disposal> OnDispose;
354             (new Regex(@"(?<begin>(?<access>(private|protected|public): )?(static )?)event
                ↪  (?<type>[a-zA-Z][:_a-zA-Z0-9]+) (?<name>[a-zA-Z][_a-zA-Z0-9]+);"),
                ↪  "${begin}Platform::Delegates::MulticastDelegate<${type}> ${name};", 0),
355             // Insert scope borders.
356             // class IgnoredExceptions { ... private: inline static std::vector<std::exception>
                ↪  _exceptionsBag;
357             // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: inline static
                ↪  std::vector<std::exception> _exceptionsBag;
358             (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n]+\r\n[\t
                    ]*{)(?<middle>((?!class).|\n)+?)(?<vectorFieldDeclaration>(?<access>(private|pro
                ↪  tected|public): )inline static std::vector<(?<argumentType>[^;\r\n]+)>
                ↪  (?<fieldName>[_a-zA-Z0-9]+);)"),
                ↪  "${classDeclarationBegin}/*~${fieldName}~*/${middle}${vectorFieldDeclaration}",
                ↪  0),
359             // Inside the scope of ~!_exceptionsBag!~ replace:
360             // _exceptionsBag.Add(exception);
361             // _exceptionsBag.push_back(exception);
362             (new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<befor
                ↪  e>((?<!/\*~\k<fieldName>~\*/)(.|\n))*?)\k<fieldName>\.Add"),
                ↪  "${scope}${separator}${before}${fieldName}.push_back", 10),
363             // Remove scope borders.
364             // /*~_exceptionsBag~*/
365             //
366             (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
367             // Insert scope borders.
368             // class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
369             // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: static std::mutex
                ↪  _exceptionsBag_mutex;
370             (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n]+\r\n[\t
                ↪  ]*{)(?<middle>((?!class).|\n)+?)(?<mutexDeclaration>private: inline static
                ↪  std::mutex (?<fieldName>[_a-zA-Z0-9]+)_mutex;)"),
                ↪  "${classDeclarationBegin}/*~${fieldName}~*/${middle}${mutexDeclaration}", 0),
371             // Inside the scope of ~!_exceptionsBag!~ replace:
372             // return std::vector<std::exception>(_exceptionsBag);
373             // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
                ↪  std::vector<std::exception>(_exceptionsBag);
374             (new Regex(@"(?<scope>/\*~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<befor
                    e>((?<!/\*~\k<fieldName>~\*/)(.|\n))*?){(?<after>((?!lock_guard)[^{};\r\n])*\k<f
                ↪  ieldName>[^;}\r\n]*;)"), "${scope}${separator}${before}{
                ↪  std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", 10),
375             // Inside the scope of ~!_exceptionsBag!~ replace:
```

```
376             // _exceptionsBag.Add(exception);
377             // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
                ↪  _exceptionsBag.Add(exception);
378             (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),
379             // Remove scope borders.
380             // /*~_exceptionsBag~*/
381             //
382             (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
383             // Insert scope borders.
384             // class IgnoredExceptions { ... public: static inline
                ↪  Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
                ↪  ExceptionIgnored = OnExceptionIgnored;
385             // class IgnoredExceptions {/*~ExceptionIgnored~*/ ... public: static inline
                ↪  Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
                ↪  ExceptionIgnored = OnExceptionIgnored;
386             (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),
387             // Inside the scope of ~!ExceptionIgnored!~ replace:
388             // ExceptionIgnored.Invoke(NULL, exception);
389             // ExceptionIgnored(NULL, exception);
390             (new Regex(@"(?<scope>/\*~(?<eventName>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before
                ↪  >((?<!/\*~\k<eventName>~\*/)(.|\n))*?)\k<eventName>\.Invoke"),
                ↪  "${scope}${separator}${before}${eventName}", 10),
391             // Remove scope borders.
392             // /*~ExceptionIgnored~*/
393             //
394             (new Regex(@"/\*~[a-zA-Z0-9]+~\*/"), "", 0),
395             // Insert scope borders.
396             // auto added = new StringBuilder();
397             // /*~sb~*/std::string added;
398             (new Regex(@"(auto|(System\.Text\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new
                ↪  (System\.Text\.)?StringBuilder\(\);"), "/*~${variable}~*/std::string
                ↪  ${variable};", 0),
399             // static void Indent(StringBuilder sb, int level)
400             // static void Indent(/*~sb~*/StringBuilder sb, int level)
401             (new Regex(@"(?<start>, |\()(System\.Text\.)?StringBuilder
                ↪  (?<variable>[a-zA-Z0-9]+)(?<end>,|\))"), "${start}/*~${variable}~*/std::string&
                ↪  ${variable}${end}", 0),
402             // Inside the scope of ~!added!~ replace:
403             // sb.ToString()
404             // sb
405             (new Regex(@"(?<scope>/\*~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.|\n)(?<before>
                ↪  ((?<!/\*~\k<variable>~\*/)(.|\n))*?)\k<variable>\.ToString\(\)"),
                ↪  "${scope}${separator}${before}${variable}", 10),
406             // sb.AppendLine(argument)
407             // sb.append(Platform::Converters::To<std::string>(argument)).append(1, '\n')
408             (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),
409             // sb.Append('\t', level);
410             // sb.append(level, '\t');
411             (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),
412             // sb.Append(argument)
413             // sb.append(Platform::Converters::To<std::string>(argument))
414             (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),
415             // Remove scope borders.
416             // /*~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(@"(?<scope>/\*method-start\*/)(?<before>((?<!/\*method-end\*/)(.|\n))*?)(?」
                ↪  <separator>[\W](?<!(::|\.|->|throw\s+)))(?<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~*/
```

```
469             (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?exception&?
                ↪  (?<variable>[_a-zA-Z0-9]+))(?<after>\W)"),
                ↪  "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
470             // Inside the scope of ~!ex!~ replace:
471             // ex.Message
472             // ex.what()
473             (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),
474             // Remove scope borders.
475             // /*~ex~*/
476             //
477             (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
478             // throw ObjectDisposedException(objectName, message);
479             // throw std::runtime_error(std::string("Attempt to access disposed object
                ↪  [").append(objectName).append("]: ").append(message).append("."));
480             (new Regex(@"throw ObjectDisposedException\((?<objectName>[a-zA-Z_][a-zA-Z0-9_]*),
                ↪  (?<message>[a-zA-Z0-9_]*[Mm]essage[a-zA-Z0-9_]*(\(\))?|[a-zA-Z_][a-zA-Z0-9_]*)\)
                ↪  ;"), "throw std::runtime_error(std::string(\"Attempt to access disposed object
                ↪  [\").append(${objectName}).append(\"]: \").append(${message}).append(\".\"));",
                ↪  0),
481             // throw ArgumentNullException(argumentName, message);
482             // throw std::invalid_argument(std::string("Argument
                ↪  ").append(argumentName).append(" is null: ").append(message).append("."));
483             (new Regex(@"throw
                ↪  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),
484             // throw ArgumentException(message, argumentName);
485             // throw std::invalid_argument(std::string("Invalid ").append(argumentName).append("
                ↪  argument: ").append(message).append("."));
486             (new Regex(@"throw
                ↪  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),
487             // throw ArgumentOutOfRangeException(argumentName, argumentValue, messageBuilder());
488             // 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("."));
489             (new Regex(@"throw ArgumentOutOfRangeException\((?<argument>[a-zA-Z]*[Aa]rgument[a-z
                A-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),
490             // throw NotSupportedException();
491             // throw std::logic_error("Not supported exception.");
492             (new Regex(@"throw NotSupportedException\(\);"), "throw std::logic_error(\"Not
                ↪  supported exception.\");", 0),
493             // throw NotImplementedException();
494             // throw std::logic_error("Not implemented exception.");
495             (new Regex(@"throw NotImplementedException\(\);"), "throw std::logic_error(\"Not
                ↪  implemented exception.\");", 0),
496             // Insert scope borders.
497             // const std::string& message
498             // const std::string& message/*~message~*/
499             (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?((std::)?string&?|char\*)
                ↪  (?<variable>[_a-zA-Z0-9]+))(?<after>\W)"),
                ↪  "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
500             // Inside the scope of /*~message~*/ replace:
501             // Platform::Converters::To<std::string>(message)
502             // message
503             (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),
504             // Remove scope borders.
505             // /*~ex~*/
506             //
```

```
            (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
            // Insert scope borders.
            // std::tuple<T, T> tuple
            // std::tuple<T, T> tuple/*~tuple~*/
            (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?tuple<[^\n]+>&?
            ↪  (?<variable>[_a-zA-Z0-9]+))(?<after>\W)"),
            ↪  "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
            // Inside the scope of ~!ex!~ replace:
            // tuple.Item1
            // std::get<1-1>(tuple)
            (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),
            // Remove scope borders.
            // /*~ex~*/
            //
            (new Regex(@"/\*~[_a-zA-Z0-9]+~\*/"), "", 0),
            // Insert scope borders.
            // class Range<T> {
            // class Range<T> {/*~type~Range<T>~*/
            (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)(template\s*<[^<>\n]*>
            ↪  )?(struct|class)
            ↪  (?<fullType>(?<typeName>[a-zA-Z0-9]+)(<[^:\n]*>)?)(\s*:\s*[^{\n]+)?[\t
            ↪  ]*(\r?\n)?[\t ]*{)"),
            ↪  "${classDeclarationBegin}/*~type~${typeName}~${fullType}~*/", 0),
            // Inside the scope of /*~type~Range<T>~*/ insert inner scope and replace:
            // public: static implicit operator std::tuple<T, T>(Range<T> range)
            // public: operator std::tuple<T, T>() const {/*~variable~Range<T>~*/
            (new Regex(@"(?<scope>/\*~type~(?<typeName>[^~\n\*]+)~(?<fullType>[^~\n\*]+)~\*/)(?<
            ↪  separator>.|\n)(?<before>((?<!/\*~type~\k<typeName>~\k<fullType>~\*/)(.|\n))*?)(
            ↪  ?<access>(private|protected|public): )static implicit operator
            ↪  (?<targetType>[^\(\n]+)\((?<argumentDeclaration>\k<fullType>
            ↪  (?<variable>[a-zA-Z0-9]+))\)(?<after>\s*\n?\s*{)"),
            ↪  "${scope}${separator}${before}${access}operator ${targetType}()
            ↪  const${after}/*~variable~${variable}~*/", 10),
            // Inside the scope of /*~type~Range<T>~*/ replace:
            // 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)); }
            // public: Range(std::tuple<T, T> tuple) : Range(std::get<1-1>(tuple),
            ↪  std::get<2-1>(tuple)) { }
            (new Regex(@"(?<scope>/\*~type~(?<typeName>[^~\n\*]+)~(?<fullType>[^~\n\*]+)~\*/)(?<
            ↪  separator>.|\n)(?<before>((?<!/\*~type~\k<typeName>~\k<fullType>~\*/)(.|\n))*?)(
            ↪  ?<access>(private|protected|public): )static implicit operator
            ↪  (\k<fullType>|\k<typeName>)\((?<arguments>[^{}\n]+)\)(\s|\n)*{(\s|\n)*return
            ↪  (new )?(\k<fullType>|\k<typeName>)\((?<passedArguments>[^\n]+)\);(\s|\n)*}"),
            ↪  "${scope}${separator}${before}${access}${typeName}(${arguments}) :
            ↪  ${typeName}(${passedArguments}) { }", 10),
            // Inside the scope of /*~variable~range~*/ replace:
            // range.Minimum
            // this->Minimum
            (new Regex(@"(?<scope>{/\*~variable~(?<variable>[^~\n]+)~\*/)(?<separator>.|\n)(?<be
            ↪  fore>(?<beforeExpression>(?<bracket>{)|(?<-bracket>})|[^{}]|\n)*?)\k<variable>\.
            ↪  (?<field>[_a-zA-Z0-9]+)(?<after>(,|;|}|
            ↪  |\))(?<afterExpression>(?<bracket>{)|(?<-bracket>})|[^{}]|\n)*?})"),
            ↪  "${scope}${separator}${before}this->${field}${after}", 10),
            // Remove scope borders.
            // /*~ex~*/
            //
            (new Regex(@"/\*~[^~\n]+~[^~\n]+~\*/"), "", 0),
            // Insert scope borders.
            // namespace Platform::Ranges { ... }
            // namespace Platform::Ranges {/*~start~namespace~Platform::Ranges~*/ ...
            ↪  /*~end~namespace~Platform::Ranges~*/}
            (new Regex(@"(?<namespaceDeclarationBegin>\r?\n(?<indent>[\t ]*)namespace
            ↪  (?<namespaceName>(?<namePart>[a-zA-Z][a-zA-Z0-9]+)(?<nextNamePart>::[a-zA-Z][a-z
            ↪  A-Z0-9]+)+)(\s|\n)*{)(?<middle>(.|\n)*)(?<end>(?<=\r?\n)\k<indent>}(?!;))"),
            ↪  "${namespaceDeclarationBegin}/*~start~namespace~${namespaceName}~*/${middle}/*~e
            ↪  nd~namespace~${namespaceName}~*/${end}",
            ↪  0),
            // Insert scope borders.
            // class Range<T> { ... };
            // class Range<T> {/*~start~type~Range<T>~T~*/ ... /*~start~type~Range<T>~T~*/};
```

```
547             (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>(.|\n)*)(?<endIndent>(?<=\r?\n)\k<indent>)(?<end>};)"),
                ↪  "${classDeclarationBegin}/*~start~type~${type}~${typeParameter}~*/${middle}${end
                ↪  Indent}/*~end~type~${type}~${typeParameter}~*/${end}",
                ↪  0),
548             // Inside scopes replace:
549             // /*~start~namespace~Platform::Ranges~*/ ... /*~start~type~Range<T>~T~*/ ...
                ↪  public: override std::int32_t GetHashCode() { return {Minimum,
                ↪  Maximum}.GetHashCode(); } ... /*~start~type~Range<T>~T~*/ ...
                ↪  /*~end~namespace~Platform::Ranges~*/
550             // /*~start~namespace~Platform::Ranges~*/ ... /*~start~type~Range<T>~T~*/ ...
                    /*~start~type~Range<T>~T~*/ ... /*~end~namespace~Platform::Ranges~*/ namespace
                ↪  std { template <typename T> struct hash<Platform::Ranges::Range<T>> {
                ↪  std::size_t operator()(const Platform::Ranges::Range<T> &obj) const { return
                ↪  {Minimum, Maximum}.GetHashCode(); } }; }
551             (new Regex(@"(?<namespaceScopeStart>/\*~start~namespace~(?<namespace>[^~\n\*]+)~\*/)
                    (?<betweenStartScopes>(.|\n)+)(?<typeScopeStart>/\*~start~type~(?<type>[^~\n\*]+
                    )~(?<typeParameter>[^~\n\*]+)~\*/)(?<before>(.|\n)+?)(?<hashMethodDeclaration>\r
                ↪  ?\n[ \t]*(?<access>(private|protected|public): )override std::int32_t
                ↪  GetHashCode\(\)(\s|\n)*{\s*(?<methodBody>[^\s][^\n]+[^\s])\s*}\s*)(?<after>(.|\n
                ↪  )+?)(?<typeScopeEnd>/\*~end~type~\k<type>~\k<typeParameter>~\*/)(?<betweenEndSco
                ↪  pes>(.|\n)+)(?<namespaceScopeEnd>/\*~end~namespace~\k<namespace>~\*/)}\r?\n"),
                ↪  "${namespaceScopeStart}${betweenStartScopes}${typeScopeStart}${before}${after}${
                ↪  typeScopeEnd}${betweenEndScopes}${namespaceScopeEnd}}" + Environment.NewLine +
                ↪  Environment.NewLine + "namespace std" + Environment.NewLine + "{" +
                ↪  Environment.NewLine + "    template <typename ${typeParameter}>" +
                ↪  Environment.NewLine + "    struct hash<${namespace}::${type}>" +
                ↪  Environment.NewLine + "    {" + Environment.NewLine + "        std::size_t
                ↪  operator()(const ${namespace}::${type} &obj) const" + Environment.NewLine + "
                ↪      {" + Environment.NewLine + "
                ↪  /*~start~method~*/${methodBody}/*~end~method~*/" + Environment.NewLine + "
                ↪   }" + Environment.NewLine + "    };" + Environment.NewLine + "}" +
                ↪  Environment.NewLine, 10),
552             // Inside scope of /*~start~method~*/ replace:
553             // /*~start~method~*/ ... Minimum ... /*~end~method~*/
554             // /*~start~method~*/ ... obj.Minimum ... /*~end~method~*/
555             (new Regex(@"(?<methodScopeStart>/\*~start~method~\*/)(?<before>.+({|,
                    ))(?<name>[a-zA-Z][a-zA-Z0-9]+)(?<after>[^\n\.\(a-zA-Z0-9]((?!/\*~end~method~\*/
                ↪  )[^\n])+)(?<methodScopeEnd>/\*~end~method~\*/)"),
                ↪  "${methodScopeStart}${before}obj.${name}${after}${methodScopeEnd}", 10),
556             // Remove scope borders.
557             // /*~start~type~Range<T>~*/
558             //
559             (new Regex(@"/\*~[^~\*\n]+(~[^~\*\n]+)*~\*/"), "", 0),
560         }.Cast<ISubstitutionRule>().ToList();
561
562         public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
563         {
564             // ICounter<int, int> c1;
565             // ICounter<int, int>* c1;
566             (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^>\r\n]+>)?)
                ↪  (?<variable>[_a-zA-Z0-9]+)(?<after> = null)?;"), "${abstractType}*
                ↪  ${variable}${after};", 0),
567             // (expression)
568             // expression
569             (new Regex(@"(\(| )\(([a-zA-Z0-9_\*:]+)\)(,| |;|\))"), "$1$2$3", 0),
570             // (method(expression))
571             // method(expression)
572             (new Regex(@"(?<firstSeparator>(\(|
                    ))\((?<method>[a-zA-Z0-9_\->\*:]+)\((?<expression>((?<parenthesis>\()|(?<-parent
                ↪  hesis>\))|[a-zA-Z0-9_\->\*:]*)+)(?(parenthesis)(?!))\)\)(?<lastSeparator>(,|
                ↪  |;|\)))"), "${firstSeparator}${method}(${expression})${lastSeparator}", 0),
573             // .append(".")
574             // .append(1, '.');
575             (new Regex(@"\.append\(""([^\\""]|\\[^""])""\)"), ".append(1, '$1')", 0),
576             // return ref _elements[node];
577             // return &_elements[node];
578             (new Regex(@"return ref ([_a-zA-Z0-9]+)\[([_a-zA-Z0-9\*]+)\];"), "return &$1[$2];",
                ↪  0),
579             // ((1, 2))
580             // ({1, 2})
581             (new Regex(@"(?<before>\(|, )\((?<first>[^\n()]+),
                ↪  (?<second>[^\n()]+)\)(?<after>\)|, )"), "${before}{${first},
                ↪  ${second}}${after}", 10),
```

```
            // {1, 2}.GetHashCode()
            // Platform::Hashing::Hash(1, 2)
            (new Regex(@"{(?<first>[^\n{}]+), (?<second>[^\n{}]+)}\.GetHashCode\(\)"),
            ↪  "Platform::Hashing::Hash(${first}, ${second})", 10),
            // range.ToString()
            // Platform::Converters::To<std::string>(range).data()
            (new Regex(@"(?<before>\W)(?<variable>[_a-zA-Z][_a-zA-Z0-9]+)\.ToString\(\)"),
            ↪  "${before}Platform::Converters::To<std::string>(${variable}).data()", 10),
            // new
            //
            (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)new\
            ↪  s+"), "${before}",
            ↪  10),
            // x == null
            // x == nullptr
            (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(?<v
            ↪  ariable>[_a-zA-Z][_a-zA-Z0-9]+)(?<operator>\s*(==|!=)\s*)null(?<after>\W)"),
            ↪  "${before}${variable}${operator}nullptr${after}", 10),
            // null
            // {}
            (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)null
            ↪  (?<after>\W)"), "${before}{}${after}",
            ↪  10),
            // default
            // 0
            (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)defa
            ↪  ult(?<after>\W)"), "${before}0${after}",
            ↪  10),
            // object x
            // void *x
            (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(?<!
            ↪  @)(object|System\.Object) (?<after>\w)"), "${before}void *${after}",
            ↪  10),
            // <object>
            // <void*>
            (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(?<!
            ↪  @)(object|System\.Object)(?<after>\W)"), "${before}void*${after}",
            ↪  10),
            // @object
            // object
            (new Regex(@"@([_a-zA-Z0-9]+)"), "$1", 0),
            // this->GetType().Name
            // typeid(this).name()
            (new Regex(@"(this)->GetType\(\)\.Name"), "typeid($1).name()", 0),
            // ArgumentNullException
            // std::invalid_argument
            (new Regex(@"(?<before>\r?\n[^""\r\n]*(""(\\""|[^""\r\n])*""[^""\r\n]*)*)(?<=\W)(Sys
            ↪  tem\.)?ArgumentNullException(?<after>\W)"),
            ↪  "${before}std::invalid_argument${after}", 10),
            // InvalidOperationException
            // std::runtime_error
            (new Regex(@"(\W)(InvalidOperationException|Exception)(\W)"),
            ↪  "$1std::runtime_error$3", 0),
            // ArgumentException
            // std::invalid_argument
            (new Regex(@"(\W)(ArgumentException|ArgumentOutOfRangeException)(\W)"),
            ↪  "$1std::invalid_argument$3", 0),
            // template <typename T> struct Range : IEquatable<Range<T>>
            // template <typename T> struct Range {
            (new Regex(@"(?<before>template <typename (?<typeParameter>[^\n]+)> (struct|class)
            ↪  (?<type>[a-zA-Z0-9]+<[^\n]+>)) : (public
            ↪  )?IEquatable<\k<type>>(?<after>(\s|\n)*{)"), "${before}${after}", 0),
            // public: delegate void Disposal(bool manual, bool wasDisposed);
            // public: delegate void Disposal(bool, bool);
            (new Regex(@"(?<before>(?<access>(private|protected|public): )delegate
                (?<returnType>[a-zA-Z][a-zA-Z0-9:]+)
            ↪  (?<delegate>[a-zA-Z][a-zA-Z0-9]+)\(((?<leftArgumentType>[a-zA-Z][a-zA-Z0-9:]+),
            ↪  )*)(?<argumentType>[a-zA-Z][a-zA-Z0-9:]+)
            ↪  (?<argumentName>[a-zA-Z][a-zA-Z0-9]+)(?<after>(,
            ↪  (?<rightArgumentType>[a-zA-Z][a-zA-Z0-9:]+)
            ↪  (?<rightArgumentName>[a-zA-Z][a-zA-Z0-9]+))*\);)"),
            ↪  "${before}${argumentType}${after}", 20),
            // public: delegate void Disposal(bool, bool);
            // using Disposal = void(bool, bool);
```

```
            (new Regex(@"(?<access>(private|protected|public): )delegate
            ↪  (?<returnType>[a-zA-Z][a-zA-Z0-9:]+)
            ↪  (?<delegate>[a-zA-Z][a-zA-Z0-9]+)\((?<argumentTypes>[^\(\)\n]*)\);"), "using
            ↪  ${delegate} = ${returnType}(${argumentTypes});", 20),
            // #region Always
            //
            (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),
            // \A \n ... namespace
            // \Anamespace
            (new Regex(@"(\A)(\r?\n)+namespace"), "$1namespace", 0),
            // \A \n ... class
            // \Aclass
            (new Regex(@"(\A)(\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(std::string args[])
    {
        printf(""Hello, world!\n"");
    }
```

```
};";
            var transformer = new CSharpToCppTransformer();
            var actualResult = transformer.Transform(helloWorldCode);
            Assert.Equal(expectedResult, actualResult);
        }
    }
}
```

Index