

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

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text.RegularExpressions;
5
6  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
7
8  namespace Platform.RegularExpressions.Transformer.CSharpToCpp
9  {
10     public class CSharpToCppTransformer : TextTransformer
11     {
12         public static readonly IList

```

```

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}${after}", 10),
66 // public ...
67 // public: ...
68 (new Regex(@"(?<newLineAndIndent>r?\n?[
    ↳ \t]*) (?<before>[^\{\\(\\r\\n)*) (?<access>private|protected|public) [ \t]+(?![^\{\\(\\r\\n)
    ↳ \n]*((?<=\\s)\\W) (interface|class|struct) (\\W) [^\{\\(\\r\\n)*[\\{\\(\\r\\n)]")",
    ↳ "${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}>${typeDefinitionEnding}{", 0),
    ↳ "public:", 0),
84 // template <typename TObject, TProperty, TValue>
85 // template <typename TObject, typename TProperty, typename TValue>
86 (new Regex(@"(?<before>template <((, )?typename [a-zA-Z0-9]+)+,
    ↳ )(?<typeParameter>[a-zA-Z0-9]+)(?<after>(,|>))", "${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);

```

```

// ~extensionMethodBuildExceptionString~*/*...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),
// private: static readonly Disposal _emptyDelegate = (manual, wasDisposed) => { };
// private: inline static std::function<Disposal> _emptyDelegate = [] (auto manual,
→ auto wasDisposed) { };
(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
→ ${secondArgument}) { };", 0),
// public: static readonly EnsureAlwaysExtensionRoot Always = new
→ EnsureAlwaysExtensionRoot();
// public: inline static EnsureAlwaysExtensionRoot Always;
(new Regex(@"(?<access>(private|protected|public): )?static readonly
→ (?<type>[a-zA-Z0-9]+(<[a-zA-Z0-9]+>)? ) (?<name>[a-zA-Z0-9_]+) = new
→ \\k<type>\\(\\);");", "{$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} =
→ ${type}({arguments});", 0),
// public: static readonly string ExceptionContentsSeparator = "---";
// public: inline static std::string ExceptionContentsSeparator = "---";
(new Regex(@"(?<access>(private|protected|public): )?(const|static readonly) string
→ (?<name>[a-zA-Z0-9_]+) = \"\"(?<string>(\\\"|\\[~\"\\r\\n]+)\"\";");", "{$access}inline
→ static std::string ${name} = \\\"${string}\\\";";", 0),
// 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 ${method} = 0;";", 0),
// TElement GetFirst();
// virtual TElement GetFirst() = 0;
(new Regex(@"(?<before>[\\r\\n]+[ ]+)(?<methodDeclaration>(?!return) [a-zA-Z0-9]+
→ [a-zA-Z0-9]+\\(\\([~\\r\\n]*\\)) (?<after>[ ]*[\\r\\n]+)");", "{$before}virtual
→ ${methodDeclaration} = 0${after}", 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();

```

```

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 $${expression}; }", 0),
160 // Count => GetSizeOrZero(Root);
161 // Count() { return GetSizeOrZero(Root); }
162 (new Regex(@"(\\W)([A-Z][a-zA-Z]+)\s+=>\s+([~;\r\n]+);"), "$1$2() { return $3; }", 0),
163 // Insert scope borders.
164 // interface IDisposable { ... }
165 // interface IDisposable { /*~start~interface~IDisposable~*/ ...
    ↳ /*~end~interface~IDisposable~*/
166 (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\\t ]*)interface[\\t
    ↳ ]*(?<type>[a-zA-Z_][a-zA-Z0-9_]*(\\<[^>\\n*>?) [~{ }]*{ } (?<middle>(\\.|\\n)*) (?<beforeE_
    ↳ nd>(\\<=\\r?\\n)\\k<indent> (?<end>})")",
    ↳ "$${classDeclarationBegin} /*~start~interface~$${type}~*/$${middle}$${beforeEnd} /*~en_
    ↳ d~interface~$${type}~*/$${end}",
    ↳ 0),
167 // Inside the scope replace:
168 // /*~start~interface~IDisposable~*/ ... bool IsDisposed { get; } ...
    ↳ /*~end~interface~IDisposable~*/
169 // /*~start~interface~IDisposable~*/ ... virtual bool IsDisposed() = 0;
    ↳ /*~end~interface~IDisposable~*/
170 (new Regex(@"(?<before>(\\<typeScopeStart>/\\*~start~interface~(?<type>[~\\n\\*]+)~\\*/)
    ↳ (\\.|\\n)+? (?<propertyDeclaration>(\\<access>(private|protected|public):
    ↳ )?(\\<propertyType>[a-zA-Z_][a-zA-Z0-9_]*<\\>*) (\\<property>[a-zA-Z_][a-zA-Z0-9_]* )
    ↳ (\\<blockOpen>[\\n\\s]*[\\{\\n\\s]*)(\\[[^\\n\\+\\+\\]\\n\\s]*)?get; (\\<blockClose>[\\n\\s]*)) (\\<
    ↳ after>(\\.|\\n)+? (\\<typeScopeEnd>/\\*~end~interface~\\k<type>~\\*/))")",
    ↳ "$${before}virtual $${propertyType} $${property}() = 0;$${after}", 20),
171 // Remove scope borders.
172 // /*~start~interface~IDisposable~*/
173 //
174 (new Regex(@"/*~[~\\*\\n]+(\\[~\\*\\n]+)*~\\*/"), "", 0),
175 // public: T Object { get; }
176 // public: const T Object;
177 (new Regex(@"(?<before>[~\\r\\r?\\n[ \\t ]*) (?<access>(private|protected|public):
    ↳ )?(\\<type>[a-zA-Z_][a-zA-Z0-9_]*<\\>*)
    ↳ (\\<property>[a-zA-Z_][a-zA-Z0-9_]*) (\\<blockOpen>[\\n\\s]*[\\{\\n\\s]*)(\\[[^\\n\\+\\+\\]\\n\\s_
    ↳ ]*)?get; (\\<blockClose>[\\n\\s]*) (\\<after>[\\n\\s]*")", "$${before}$${access}const
    ↳ $${type} $${property};$${after}", 2),
178 // public: bool IsDisposed { get => _disposed > 0; }
179 // public: bool IsDisposed() { return _disposed > 0; }

```

```

(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) [\n\s]
→ ]*) ?get\s*=>\s*(?<expression>[^\n]+); (?<blockClose>[\n\s]*{[\n\s]*})",
→ "${before}${access}${virtual}bool ${property}() ${blockOpen}return
→ ${expression}; ${blockClose}", 2),
// protected: virtual std::string ObjectName { get => GetType().Name; }
// protected: virtual std::string ObjectName() { return GetType().Name; }
(new Regex(@"(?<before>[^\r]\r?\n[ \t]*) (?<access>(private|protected|public):
) ?(?<virtual>virtual ) ?(?<type>[a-zA-Z_][a-zA-Z0-9_<:;>]*)
→ (?<property>[a-zA-Z_][a-zA-Z0-9_]*) (?<blockOpen>[\n\s]*{[\n\s]*} ([^\n]+\n) [\n\s]
→ ]*) ?get\s*=>\s*(?<expression>[^\n]+); (?<blockClose>[\n\s]*{[\n\s]*})",
→ "${before}${access}${virtual}${type} ${property}() ${blockOpen}return
→ ${expression}; ${blockClose}", 2),
// ArgumentInRange(string message) { string messageBuilder() { return message; }
// ArgumentInRange(string message) { auto messageBuilder = [&]() -> string { return
→ message; };
(new Regex(@"(?<before>\W[_a-zA-Z0-9]+\n((^\n)\n)*[\s\n]*{[\s\n]*{[^\n]+\n)*?(\r?\n)
→ ?[ \t]*) (?<returnType>[_a-zA-Z0-9*:] +[_a-zA-Z0-9*:] *)
→ (?<methodName>[_a-zA-Z0-9+] +)\n((?<arguments>[^\n]\n)*)\n\s*{(?<body>("[^"\n]+"" |
→ [^}] | \n)+?)})", "${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()> $2", 0),
// Action<TElement> free
// std::function<void(TElement)> free
(new Regex(@"Action<(?<typeParameters>[a-zA-Z0-9+] +,
→ ([a-zA-Z0-9+] +)*)> (?<after>| (?<variable>[a-zA-Z0-9+] +))",
→ "std::function<void(${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 \*+]+)\\n((^\n)\r\n)+?)", "$1$2 override", 0),
// return {range.Minimum, range.Maximum}
// return {range.Minimum, range.Maximum}
(new Regex(@"(?<before>return\s*)\\n((?<values>[^\n]\n)+)\\n(?!\n) (?<after>\W)",
→ "${before}${values}} ${after}", 0),
// string
// std::string
(new Regex(@"(?<before>\W) (?<!:):)string(?<after>\W)",
→ "${before}std::string${after}", 0),
// System.ValueTuple
// std::tuple
(new Regex(@"(?<before>\W) (System\\.)?ValueTuple(?!\s*=\n) (?<after>\W)",
→ "${before}std::tuple${after}", 0),
// sbyte
// std::int8_t
(new Regex(@"(?<before>\W) ((System\\.)?SB|sb)yte(?!\s*=\n) (?<after>\W)",
→ "${before}std::int8_t${after}", 0),
// short
// std::int16_t
(new Regex(@"(?<before>\W) ((System\\.)?Int16|short) (?!\s*=\n) (?<after>\W)",
→ "${before}std::int16_t${after}", 0),
// int
// std::int32_t

```

```

231 (new Regex(@"(?<before>\W)((System\.)?I|i)nt(32)?(?!\s*=\|()\ (?<after>\W)"),
232     ↳ "${before}std::int32_t${after}", 0),
233 // long
234 (new Regex(@"(?<before>\W)((System\.)?Int64|long)(?! \s*=\|()\ (?<after>\W)"),
235     ↳ "${before}std::int64_t${after}", 0),
236 // byte
237 (new Regex(@"(?<before>\W)((System\.)?Byte|byte)(?! \s*=\|()\ (?<after>\W)"),
238     ↳ "${before}std::uint8_t${after}", 0),
239 // ushort
240 (new Regex(@"(?<before>\W)((System\.)?UInt16|ushort)(?! \s*=\|()\ (?<after>\W)"),
241     ↳ "${before}std::uint16_t${after}", 0),
242 // uint
243 (new Regex(@"(?<before>\W)((System\.)?UI|ui)nt(32)?(?!\s*=\|()\ (?<after>\W)"),
244     ↳ "${before}std::uint32_t${after}", 0),
245 // ulong
246 (new Regex(@"(?<before>\W)((System\.)?UInt64|ulong)(?! \s*=\|()\ (?<after>\W)"),
247     ↳ "${before}std::uint64_t${after}", 0),
248 // char*[] args
249 (new Regex(@"([_a-zA-Z0-9:\*]?)\[\] ([a-zA-Z0-9]+)", "$1 $2[]", 0),
250 // float.MinValue
251 (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MinValue(?<after>\W|
252     ↳ )"), "${before}std::numeric_limits<${type}>::lowest()${after}",
253     ↳ 0),
254 // double.MaxValue
255 (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+|float|double)\.MaxValue(?<after>\W|
256     ↳ )"), "${before}std::numeric_limits<${type}>::max()${after}",
257     ↳ 0),
258 // using Platform.Numbers;
259 //
260 (new Regex(@"([\r\n]{2}|^)\s*?using [\a-zA-Z0-9+;\s*?${")", "", 0),
261 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
262 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
263 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(<[a-zA-Z0-9 ,]+>)? : ([a-zA-Z0-9]+)",
264     ↳ "$1 $2$3 : public $4", 0),
265 // System.IDisposable
266 // System::IDisposable
267 (new Regex(@"(?<before>System(::[a-zA-Z_]\w*)*)\. (?<after>[a-zA-Z_]\w*)"),
268     ↳ "${before}::${after}", 20),
269 // class IProperty : ISetter<TValue, TObjct>, IProvider<TValue, TObjct>
270 // class IProperty : public ISetter<TValue, TObjct>, public IProvider<TValue,
271     ↳ TObjct>
272 (new Regex(@"(?<before>(interface|struct|class) [a-zA-Z_]\w* : ((public
273     ↳ [a-zA-Z_][\w:]*(<[a-zA-Z0-9 ,]+>)?,
274     ↳ )+)?(?<inheritedType>(?!public)[a-zA-Z_][\w:]*(<[a-zA-Z0-9 ,]+>)?(?<after>(,
275     ↳ [a-zA-Z_][\w:]*(!>)|[\r\n]+)))", "${before}public ${inheritedType}${after}",
276     ↳ 10),
277 // interface IDisposable {
278 // class IDisposable { public:
279 (new Regex(@"(?<before>\r?\n)(?<indent>[ \t]*)interface
280     ↳ (?<interface>[a-zA-Z_]\w*)(?<typeDefinitionEnding>[~{]+){")",
281     ↳ "${before}${indent}class ${interface}${typeDefinitionEnding}{\" +
282     ↳ Environment.NewLine + "      public:", 0),
283 // struct TreeElement { }
284 // struct TreeElement { };
285 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([~;])", "$1
286     ↳ $2$3{$4};$5", 0),
287 // class Program { }
288 // class Program { };
289 (new Regex(@"(?<type>struct|class)
290     ↳ (?<name>[a-zA-Z0-9]+[~\r\n]*) (?<beforeBody>[\r\n]+(?<indentLevel>[ \t
291     ↳ ]*)?)\{(?<body>[ \S\s]+?[ \r\n]+\k<indentLevel>)\}(?<afterBody>[~;]|$)", "${type}
292     ↳ ${name}${beforeBody}${body}};${afterBody}", 0),
293 // Insert scope borders.
294 // ref TElement root
295 // ~!root!~ref TElement root
296 (new Regex(@"(?<definition>(?! = \|()\ (ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<ref>))
297     ↳ (?<variable>[a-zA-Z0-9]+)(?=\|, \| =))", "~!${variable}!~${definition}", 0),
298 // Inside the scope of ~!root!~ replace:

```

```

282 // root
283 // *root
284 (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),
285 // Remove scope borders.
286 // ~!root!~
287 //
288 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", 5),
289 // ref auto root = ref
290 // ref auto root =
291 (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)"), "$1* $2 =$3", 0),
292 // *root = ref left;
293 // root = left;
294 (new Regex(@"\*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", 0),
295 // (ref left)
296 // (left)
297 (new Regex(@"\ (ref ([a-zA-Z0-9]+)(\)|\(|,))"), "($1$2", 0),
298 // ref TElement
299 // TElement*
300 (new Regex(@"( |\\())ref ([a-zA-Z0-9]+) "), "$1$2* ", 0),
301 // ref sizeBalancedTree.Root
302 // &sizeBalancedTree->Root
303 (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)"), "&$1->$2", 0),
304 // ref GetElement(node).Right
305 // &GetElement(node)->Right
306 (new Regex(@"ref ([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+)\)\)\.([a-zA-Z0-9]+)"),
→ "&$1($2)->$3", 0),
307 // GetElement(node).Right
308 // GetElement(node)->Right
309 (new Regex(@"([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+)\)\)\.([a-zA-Z0-9]+)"), "$1($2)->$3", 0),
310 // [Fact]npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
311 // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
312 (new Regex(@"\[Fact\\]\[s\\n\]+(public:)?(static)?void ([a-zA-Z0-9]+)\\(\\)"), "public:
→ TEST_METHOD($3)", 0),
313 // class TreesTests
314 // TEST_CLASS(TreesTests)
315 (new Regex(@"class ([a-zA-Z0-9]+Tests)"), "TEST_CLASS($1)", 0),
316 // Assert.Equal
317 // Assert::AreEqual
318 (new Regex(@"(?<type>Assert)\\.(<method>(Not)?Equal)"), "${type}::Are${method}", 0),
319 // Assert.Throws
320 // Assert::ExpectException
321 (new Regex(@"(Assert)\\.Throws"), "$1::ExpectException", 0),
322 // Assert.True
323 // Assert::IsTrue
324 (new Regex(@"(Assert)\\. (True|False)"), "$1::Is$2", 0),
325 // $"Argument {argumentName} is null."
326 // std::string("Argument
→ ").append(Platform::Converters::To<std::string>(argumentName)).append(" is
→ null.")
327 (new Regex(@"\$""(?<left>(\\""| [^""\r\n])*){(?<expression>[_a-zA-Z0-9]+)}(?<right>(\\
→ ""| [^""\r\n])*)"",
→ "std::string(\$\"${left}\").append(Platform::Converters::To<std::string>(${expres
→ sion})).append(\$\"${right}\")",
→ 10),
328 // $"
329 // "
330 (new Regex(@"\$""""), "\"", 0),
331 // std::string(std::string("[").append(Platform::Converters::To<std::string>(Minimum)
→ )).append(",
→ ").append(Platform::Converters::To<std::string>(Maximum)).append("]")
332 // std::string("[").append(Platform::Converters::To<std::string>(Minimum)).append(",
→ ").append(Platform::Converters::To<std::string>(Maximum)).append("]")
333 (new Regex(@"std::string\\((?<begin>std::string\\(""(\\""| [^""])*)""\\)\\.append\\((Platf
→ orm::Converters::To<std::string>\\([~]\\n\\+\\)| [~]\\n\\+\\))\\+\\)\\)\\.append"),
→ "${begin}.append", 10),
334 // Console.WriteLine("...")
335 // printf("...\n")
336 (new Regex(@"Console\\.WriteLine\\(""([~""\r\n]+)""\\)"), "printf(\$\"$1\\n\\)", 0),
337 // TElement Root;
338 // TElement Root = 0;
339 (new Regex(@"(<before>\r?\n[t ]+)(?<access>(private|protected|public)(:
→ )?)?(?<type>[a-zA-Z0-9:~]+(?<!return)) (?<name>[_a-zA-Z0-9~]+);"),
→ "${before}${access}${type} ${name} = 0;", 0),

```



```

340 // TreeElement _elements[N];
341 // TreeElement _elements[N] = { {0} };
342 (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),
343 // auto path = new TElement[MaxPath];
344 // TElement path[MaxPath] = { {0} };
345 (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[$4] = { {0} };", 0),
346 // bool Equals(Range<T> other) { ... }
347 // bool operator ==(const Key &other) const { ... }
348 (new Regex(@"(?<before>\\r?\\n[\\t ]+bool )Equals\\((?<type>[\\n]+)
    ↳ (?<variable>[a-zA-Z0-9]+)\\)(?<after>(\\s|\\n)*\\)") , "${before}operator ==(const
    ↳ ${type} &${variable}) const${after}", 0),
349 // Insert scope borders.
350 // class Range { ... public: override std::string ToString() { return ...; }
351 // class Range { /*~Range<T>~*/ ... public: override std::string ToString() { return
    ↳ ...; }
352 (new Regex(@"(?<classDeclarationBegin>\\r?\\n(?<indent>[\\t ]*)template <typename
    ↳ (?<typeParameter>[~<>\\n]+> (struct|class)
    ↳ (?<type>[a-zA-Z0-9]+<\\k<typeParameter>>)(\\s*:\\s*[~{\\n]+)?[\\t ]*(\\r?\\n)?[\\t
    ↳ ]*(?<middle>((?!class|struct)\\.\\n)+?)?(?<toStringDeclaration>(\\s*<access>(private|
    ↳ |protected|public): )override std::string ToString\\(\\))") ,
    ↳ "${classDeclarationBegin}/*~${type}~*/${middle}${toStringDeclaration}", 0),
353 // Inside the scope of ~!Range!~ replace:
354 // public: override std::string ToString() { return ...; }
355 // public: operator std::string() const { return ...; }\\n\\npublic: friend
    ↳ std::ostream & operator <<(std::ostream &out, const A &obj) { return out <<
    ↳ (std::string)obj; }
356 (new Regex(@"(?<scope>\\/\\s*(?<type>[a-zA-Z0-9<>:]+)~\\/)(?<separator>\\.\\n)(?<before>
    ↳ ((?!\\/\\s*~\\k<type>~\\/)(\\.\\n)*)?(?<toStringDeclaration>\\r?\\n(?<indent>[
    ↳ \\t]*)?(?<access>(private|protected|public): )override std::string ToString\\(\\)
    ↳ (?<toStringMethodBody>[~}\\n]+))") , "${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
    ↳ ${type} &obj) { return out << (std::string)obj; }", 0),
357 // Remove scope borders.
358 // /*~Range~*/
359 //
360 (new Regex(@"\\/\\s*[a-zA-Z0-9<>:]+~\\/") , "", 0),
361 // private: inline static ConcurrentBag<std::exception> _exceptionsBag;
362 // private: inline static std::mutex _exceptionsBag_mutex; \\n\\n private: inline
    ↳ static std::vector<std::exception> _exceptionsBag;
363 (new Regex(@"(?<begin>\\r?\\n?(?<indent>[\\t ]+))?(?<access>(private|protected|public):
    ↳ )?inline static ConcurrentBag<(\\s*<argumentType>[~;\\r\\n]+)>
    ↳ (?<name>[_a-zA-Z0-9]+);") , "${begin}private: inline static std::mutex
    ↳ ${name}_mutex;" + Environment.NewLine + Environment.NewLine +
    ↳ "${indent}${access}inline static std::vector<${argumentType}> ${name};", 0),
364 // public: static IReadonlyCollection<std::exception> GetCollectedExceptions() {
    ↳ return _exceptionsBag; }
365 // public: static std::vector<std::exception> GetCollectedExceptions() { return
    ↳ std::vector<std::exception>(_exceptionsBag); }
366 (new Regex(@"(?<access>(private|protected|public): )?static
    ↳ IReadonlyCollection<(\\s*<argumentType>[~;\\r\\n]+)> (?<methodName>[_a-zA-Z0-9]+)\\(\\)
    ↳ { return (?<fieldName>[_a-zA-Z0-9]+); }") , "${access}static
    ↳ std::vector<${argumentType}> ${methodName}() { return
    ↳ std::vector<${argumentType}>({${fieldName}}); }", 0),
367 // public: static event EventHandler<std::exception> ExceptionIgnored =
    ↳ OnExceptionIgnored; ... };
368 // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
    ↳ const std::exception&> ExceptionIgnored = OnExceptionIgnored; };
369 (new Regex(@"(?<begin>\\r?\\n(\\r?\\n)?(?<halfIndent>[
    ↳ \\t]+)\\k<halfIndent>)(?<access>(private|protected|public): )?static event
    ↳ EventHandler<(\\s*<argumentType>[~;\\r\\n]+)> (?<name>[_a-zA-Z0-9]+) = (?<defaultDele
    ↳ gate>[_a-zA-Z0-9]+);(?<middle>(\\.\\n)+?)?(?<end>\\r?\\n\\k<halfIndent>});") ,
    ↳ "${middle}" + Environment.NewLine + Environment.NewLine +
    ↳ "${halfIndent}${halfIndent}${access}static inline
    ↳ Platform::Delegates::MulticastDelegate<void(void*, const ${argumentType}&>
    ↳ ${name} = ${defaultDelegate};${end}", 0),
370 // public: event Disposal OnDispose;
371 // public: Platform::Delegates::MulticastDelegate<Disposal> OnDispose;
372 (new Regex(@"(?<begin>(\\s*<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),
373 // Insert scope borders.

```



```

374 // class IgnoredExceptions { ... private: inline static std::vector<std::exception>
    ↳ _exceptionsBag;
375 // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: inline static
    ↳ std::vector<std::exception> _exceptionsBag;
376 (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]+);)");
    ↳ "$${classDeclarationBegin}/*~${fieldName}~*/${middle}${vectorFieldDeclaration}",
    ↳ 0),
377 // Inside the scope of ~!_exceptionsBag!~ replace:
378 // _exceptionsBag.Add(exception);
379 // _exceptionsBag.push_back(exception);
380 (new Regex(@"(?<scope>/\s*(?<fieldName>[_a-zA-Z0-9]+)~\s*/)(?<separator>.\|\n)(?<before>
    ↳ e>((?!/\s*\k<fieldName>~\s*/)(.\|\n))*?)\k<fieldName>\.Add"),
    ↳ "$${scope}${separator}${before}${fieldName}.push_back", 10),
381 // Remove scope borders.
382 // /*~_exceptionsBag~*/
383 //
384 (new Regex(@"/\s*~[_a-zA-Z0-9]+~\s*/"), "", 0),
385 // Insert scope borders.
386 // class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
387 // class IgnoredExceptions {/*~_exceptionsBag~*/ ... private: static std::mutex
    ↳ _exceptionsBag_mutex;
388 (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;);"),
    ↳ "$${classDeclarationBegin}/*~${fieldName}~*/${middle}${mutexDeclaration}", 0),
389 // Inside the scope of ~!_exceptionsBag!~ replace:
390 // return std::vector<std::exception>(_exceptionsBag);
391 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
    ↳ std::vector<std::exception>(_exceptionsBag);
392 (new Regex(@"(?<scope>/\s*(?<fieldName>[_a-zA-Z0-9]+)~\s*/)(?<separator>.\|\n)(?<before>
    ↳ e>((?!/\s*\k<fieldName>~\s*/)(.\|\n))*?){(?<after>((?!lock_guard)[^{};\r\n])*\k<f
    ↳ ieldName>[~;}\r\n]*;);"), "$${scope}${separator}${before}{
    ↳ std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", 10),
393 // Inside the scope of ~!_exceptionsBag!~ replace:
394 // _exceptionsBag.Add(exception);
395 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
    ↳ _exceptionsBag.Add(exception);
396 (new Regex(@"(?<scope>/\s*(?<fieldName>[_a-zA-Z0-9]+)~\s*/)(?<separator>.\|\n)(?<before>
    ↳ e>((?!/\s*\k<fieldName>~\s*/)(.\|\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),
397 // Remove scope borders.
398 // /*~_exceptionsBag~*/
399 //
400 (new Regex(@"/\s*~[_a-zA-Z0-9]+~\s*/"), "", 0),
401 // Insert scope borders.
402 // class IgnoredExceptions { ... public: static inline
    ↳ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
    ↳ ExceptionIgnored = OnExceptionIgnored;
403 // class IgnoredExceptions {/*~ExceptionIgnored~*/ ... public: static inline
    ↳ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
    ↳ ExceptionIgnored = OnExceptionIgnored;
404 (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]+);)");
    ↳ "$${classDeclarationBegin}/*~${name}~*/${middle}${eventDeclaration}", 0),
405 // Inside the scope of ~!ExceptionIgnored!~ replace:
406 // ExceptionIgnored.Invoke(NULL, exception);
407 // ExceptionIgnored(NULL, exception);
408 (new Regex(@"(?<scope>/\s*(?<eventName>[_a-zA-Z0-9]+)~\s*/)(?<separator>.\|\n)(?<before>
    ↳ >((?!/\s*\k<eventName>~\s*/)(.\|\n))*?)\k<eventName>\.Invoke"),
    ↳ "$${scope}${separator}${before}${eventName}", 10),
409 // Remove scope borders.
410 // /*~ExceptionIgnored~*/
411 //
412 (new Regex(@"/\s*~[_a-zA-Z0-9]+~\s*/"), "", 0),
413 // Insert scope borders.
414 // auto added = new StringBuilder();
415 // /*~sb~*/std::string added;

```

```

416 (new Regex(@"(auto|(System\.Text\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new
    ↳ (System\.Text\.)?StringBuilder\(\);"), "/~*${variable}~/std::string
    ↳ ${variable};", 0),
417 // static void Indent(StringBuilder sb, int level)
418 // static void Indent(/*~sb~/StringBuilder sb, int level)
419 (new Regex(@"(?<start>, |\() (System\.Text\.)?StringBuilder
    ↳ (?<variable>[a-zA-Z0-9]+) (?<end>, |\))"), "${start}/*~*${variable}~/std::string&
    ↳ ${variable}${end}", 0),
420 // Inside the scope of ~!added!~ replace:
421 // sb.ToString()
422 // sb
423 (new Regex(@"(?<scope>/\~* (?<variable>[a-zA-Z0-9]+) ~\~/) (?<separator>.\|\\n) (?<before>
    ↳ ((?!/\~*~\k<variable>~\~/) (. \|\\n))*) \k<variable>\.ToString\(\)"),
    ↳ "${scope}${separator}${before}${variable}", 10),
424 // sb.AppendLine(argument)
425 // sb.append(Platform::Converters::To<std::string>(argument)).append(1, '\\n')
426 (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),
427 // sb.Append('\\t', level);
428 // sb.append(level, '\\t');
429 (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),
430 // sb.Append(argument)
431 // sb.append(Platform::Converters::To<std::string>(argument))
432 (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),
433 // Remove scope borders.
434 // /*~sb~/
435 //
436 (new Regex(@"/*~*[a-zA-Z0-9]+~*/"), "", 0),
437 // Insert scope borders.
438 // auto added = new HashSet<TElement>();
439 // ~!added!~std::unordered_set<TElement> added;
440 (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
    ↳ HashSet<(?<element>[a-zA-Z0-9]+)>\(\);"),
    ↳ "~!${variable}!~std::unordered_set<${element}> ${variable};", 0),
441 // Inside the scope of ~!added!~ replace:
442 // added.Add(node)
443 // added.insert(node)
444 (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),
445 // Inside the scope of ~!added!~ replace:
446 // added.Remove(node)
447 // added.erase(node)
448 (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(${argument})", 10),
449 // if (added.insert(node)) {
450 // if (!added.contains(node)) { added.insert(node);
451 (new Regex(@"if \\((?<variable>[a-zA-Z0-9]+)\\.insert\\((?<argument>[a-zA-Z0-9]+)\\)\\) (?
    ↳ <separator>[\\t ]*[\\r\\n]+) (?<indent>[\\t ]*){", "if
    ↳ (!${variable}.contains(${argument})) ${separator}${indent}{ " +
    ↳ Environment.NewLine + "${indent}    ${variable}.insert(${argument});", 0),
452 // Remove scope borders.
453 // ~!added!~
454 //
455 (new Regex(@"~![a-zA-Z0-9]+!~"), "", 5),
456 // Insert scope borders.
457 // auto random = new System::Random(0);
458 // std::srand(0);
459 (new Regex(@"[a-zA-Z0-9\\.]+ ([a-zA-Z0-9]+) = new
    ↳ (System::)?Random\\((([a-zA-Z0-9]+)\\)");", "~!$1!~std::srand($3);", 0),
460 // Inside the scope of ~!random!~ replace:
461 // random.Next(1, N)
462 // (std::rand() % N) + 1

```

```

463 (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),
464 // Remove scope borders.
465 // ~!random!~
466 //
467 (new Regex(@"~![a-zA-Z0-9]+!~"), "", 5),
468 // Insert method body scope starts.
469 // void PrintNodes(TElement node, StringBuilder sb, int level) {
470 // void PrintNodes(TElement node, StringBuilder sb, int level) {/*method-start*/
471 (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*/${end}}",
    ↳ 0),
472 // Insert method body scope ends.
473 // {/*method-start*/...}
474 // {/*method-start*/.../*method-end*/}
475 (new Regex(@"{/{/*method-start*//(?<body>((?<bracket>\\{)|(?!-bracket>\\})|[^\\{\\}])*+)"
    ↳ \\})", "{/{/*method-start*/${body}/*method-end*/}",
    ↳ 0),
476 // Inside method bodies replace:
477 // GetFirst(
478 // this->GetFirst(
479 (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),
480 // Remove scope borders.
481 // /*method-start*/
482 //
483 (new Regex(@"/*method-(start|end)\\*/"), "", 0),
484 // Insert scope borders.
485 // const std::exception& ex
486 // const std::exception& ex/*~ex~*/
487 (new Regex(@"(?<before>\\(| )(?<variableDefinition>(const )?(std::)?exception&?
    ↳ (?<variable>[_a-zA-Z0-9]+)) (?<after>\\W)"),
    ↳ "${before}${variableDefinition}/*~${variable}~/*${after}", 0),
488 // Inside the scope of ~!ex!~ replace:
489 // ex.Message
490 // ex.what()
491 (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),
492 // Remove scope borders.
493 // /*~ex~*/
494 //
495 (new Regex(@"/*~[_a-zA-Z0-9]+~\\*/"), "", 0),
496 // throw ObjectDisposedException(objectName, message);
497 // throw std::runtime_error(std::string("Attempt to access disposed object
    ↳ ").append(objectName).append(": ").append(message).append("."));
498 (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),
499 // throw ArgumentNullException(argumentName, message);
500 // throw std::invalid_argument(std::string("Argument
    ↳ ").append(argumentName).append(" is null: ").append(message).append("."));
501 (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),
502 // throw ArgumentException(message, argumentName);
503 // throw std::invalid_argument(std::string("Invalid ").append(argumentName).append("
    ↳ argument: ").append(message).append("."));
504 (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),

```

```

505 // throw ArgumentOutOfRangeException(argumentName, argumentValue, messageBuilder());
506 // 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("."););
507 (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),
508 // throw NotSupportedException();
509 // throw std::logic_error("Not supported exception.");
510 (new Regex(@"throw NotSupportedException\(\);)", "throw std::logic_error(\"Not
    ↳ supported exception.\");", 0),
511 // throw NotImplementedException();
512 // throw std::logic_error("Not implemented exception.");
513 (new Regex(@"throw NotImplementedException\(\);)", "throw std::logic_error(\"Not
    ↳ implemented exception.\");", 0),
514 // Insert scope borders.
515 // const std::string& message
516 // const std::string& message/*~message~/
517 (new Regex(@"(?<before>\(|\s)(?<variableDefinition>(const\s)?((std::)?string&?|char\*)
    ↳ (?<variable>[_a-zA-Z0-9]+)))(?<after>\W)",
    ↳ "${before}${variableDefinition}/*~${variable}~/${after}", 0),
518 // Inside the scope of /*~message~/ replace:
519 // Platform::Converters::To<std::string>(message)
520 // message
521 (new Regex(@"(?<scope>\/\s*(?<variable>[_a-zA-Z0-9]+)~\s*)(?<separator>.\|\n)(?<before>
    ↳ >((?!\/\s*\k<variable>~\s*)(.\|\n))*?)Platform::Converters::To<std::string>\(\k<v
    ↳ ariable>\)", "${scope}${separator}${before}${variable}",
    ↳ 10),
522 // Remove scope borders.
523 // /*~ex~/
524 //
525 (new Regex(@"\/\s*[_a-zA-Z0-9]+~\s\/", "", 0),
526 // Insert scope borders.
527 // std::tuple<T, T> tuple
528 // std::tuple<T, T> tuple/*~tuple~/
529 (new Regex(@"(?<before>\(|\s)(?<variableDefinition>(const\s)?(std::)?tuple<[^\n]+&?
    ↳ (?<variable>[_a-zA-Z0-9]+)))(?<after>\W)",
    ↳ "${before}${variableDefinition}/*~${variable}~/${after}", 0),
530 // Inside the scope of ~!ex!~ replace:
531 // tuple.Item1
532 // std::get<1-1>(tuple)
533 (new Regex(@"(?<scope>\/\s*(?<variable>[_a-zA-Z0-9]+)~\s*)(?<separator>.\|\n)(?<before>
    ↳ >((?!\/\s*\k<variable>~\s*)(.\|\n))*?)\k<variable>\.Item(?<itemNumber>\d+)(?<afte
    ↳ r>\W)",
    ↳ "${scope}${separator}${before}std::get<${itemNumber}-1>(${variable})${after}",
    ↳ 10),
534 // Remove scope borders.
535 // /*~ex~/
536 //
537 (new Regex(@"\/\s*[_a-zA-Z0-9]+~\s\/", "", 0),
538 // Insert scope borders.
539 // class Range<T> {
540 // class Range<T> {/*~type~Range<T>~/
541 (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)(template\s*<[^\<>\n]*>
    ↳ )?(struct|class)
    ↳ (?<fullType>(?(type)Name[a-zA-Z0-9]+)(<[^\n]*>)?)(\s*:\s*[^\n]+)?[\t
    ↳ ]*(\r?\n)?[\t ]*{)"),
    ↳ "${classDeclarationBegin}/*~type~${typeName}~${fullType}~/", 0),
542 // Inside the scope of /*~type~Range<T>~/ insert inner scope and replace:
543 // public: static implicit operator std::tuple<T, T>(Range<T> range)
544 // public: operator std::tuple<T, T>() const {/*~variable~Range<T>~/
545 (new Regex(@"(?<scope>\/\s*(?<type>Name[a-zA-Z0-9]+)~\s*(?<fullType>[^\n]*~\s\/)(?<
    ↳ separator>.\|\n)(?<before>((?!\/\s*(?<type>Name[a-zA-Z0-9]+)~\s*(?<fullType>[^\n]*~\s\/)(.\|\n))*?)
    ↳ (?<access>(private|protected|public): )static implicit operator
    ↳ (?<targetType>[^\n]*~\s\/)\(((?<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:

```

```

547 // 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)); }
548 // public: Range(std::tuple<T, T> tuple) : Range(std::get<1-1>(tuple),
    ↳ std::get<2-1>(tuple)) { }
549 (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),
550 // Inside the scope of /*~variable~range~/ replace:
551 // range.Minimum
552 // this->Minimum
553 (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}${after}", 10),
554 // Remove scope borders.
555 // /*~ex~/
556 //
557 (new Regex(@"\/\~[~\n\*]+[~\n\*]+~\*/"), "", 0),
558 // Insert scope borders.
559 // namespace Platform::Ranges { ... }
560 // namespace Platform::Ranges {/*~start~namespace~Platform::Ranges~/ ...
    ↳ /*~end~namespace~Platform::Ranges~/}
561 (new Regex(@"(?<namespaceDeclarationBegin>\r?\n(?<indent>[\t ]*)namespace
    ↳ (?<namespaceName>(?(namePart>[a-zA-Z][a-zA-Z0-9]+)(?(nextNamePart>:[a-zA-Z][a-z
    ↳ A-Z0-9]+))(\s|\n)*)(?(middle>(\.|\n)*)(?(end>(?(<=\r?\n)\k<indent>}{?!;))"),
    ↳ "${namespaceDeclarationBegin}/*~start~namespace~${namespaceName}~/${middle}/*~e
    ↳ nd~namespace~${namespaceName}~/${end}",
    ↳ 0),
562 // Insert scope borders.
563 // class Range<T> { ... };
564 // class Range<T> {/*~start~type~Range<T>~T~/ ... /*~end~type~Range<T>~T~/};
565 (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>){!(;))"),
    ↳ "${classDeclarationBegin}/*~start~type~${type}~${typeParameter}~/${middle}${end}
    ↳ Indent}/*~end~type~${type}~${typeParameter}~/${end}",
    ↳ 0),
566 // Inside the scope replace:
567 // /*~start~namespace~Platform::Ranges~/ ... /*~start~type~Range<T>~T~/ ...
    ↳ public: override std::int32_t GetHashCode() { return {Minimum,
    ↳ Maximum}.GetHashCode(); } ... /*~end~type~Range<T>~T~/ ...
    ↳ /*~end~namespace~Platform::Ranges~/
568 // /*~start~namespace~Platform::Ranges~/ ... /*~start~type~Range<T>~T~/ ...
    ↳ /*~end~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(); } }; }
569 (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>[~\n\*]+[~\n\*])\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),
570 // Inside scope of /*~start~method~/ replace:
571 // /*~start~method~/ ... Minimum ... /*~end~method~/
572 // /*~start~method~/ ... obj.Minimum ... /*~end~method~/

```

[illegible]


```
// Inside the scope replace:
// /*~app-domain~_currentDomain~/ ... _currentDomain.ProcessExit += OnProcessExit;
// /*~app-domain~_currentDomain~/ ... std::atexit(OnProcessExit);
(new Regex(@"(?<before>(?(fieldScopeStart)/\~*app-domain~(?(field)[^~\n]*+)^~*/)(.|\n)+?)\k<field>\.ProcessExit[\t ]*\+=[\t]*(?(eventHandler>[a-zA-Z_][a-zA-Z0-9_]*)");), "${before}std::atexit(${eventHandler});/*~process-exit-handler~${eventHandler}~*/", 20),
// Inside the scope replace:
// /*~app-domain~_currentDomain~/ ... _currentDomain.ProcessExit -= OnProcessExit;
// /*~app-domain~_currentDomain~/ ... /* No translation. It is not possible to unsubscribe from std::atexit. */
(new Regex(@"(?<before>(?(fieldScopeStart)/\~*app-domain~(?(field)[^~\n]*+)^~*/)(.|\n)+?r?\n[\t ]*\k<field>\.ProcessExit[\t ]*\-=[\t]*(?(eventHandler>[a-zA-Z_][a-zA-Z0-9_]*)");), "${before}/* No translation. It is not possible to unsubscribe from std::atexit. */", 20),
// Inside the scope replace:
// /*~process-exit-handler~OnProcessExit~/ ... static void OnProcessExit(void *sender, EventArgs e)
// /*~process-exit-handler~OnProcessExit~/ ... static void OnProcessExit()
(new Regex(@"(?<before>(?(fieldScopeStart)/\~*process-exit-handler~(?(handler)[^~\n\*]+)^~*/)(.|\n)+?static[\t ]+void[\t ]+\k<handler>(\(|^()\n]+\))", "${before})", 20),
// Remove scope borders.
// /*~app-domain~_currentDomain~/
//
(new Regex(@"\/\~*[^~\*\n]+(~[^~\*\n]*)*\~*/"), "", 0),
// AppDomain.CurrentDomain.ProcessExit -= OnProcessExit;
// /* No translation. It is not possible to unsubscribe from std::atexit. */
(new Regex(@"AppDomain\.CurrentDomain\.ProcessExit -= ([a-zA-Z_][a-zA-Z0-9_]*)");),
    "/* No translation. It is not possible to unsubscribe from std::atexit. */", 0),
}.Cast<ISubstitutionRule>().ToList();

public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
{
    // IDisposable disposable)
    // IDisposable &disposable)
    (new Regex(@"(?<argumentAbstractType>I[A-Z][a-zA-Z0-9]+(<[^\r\n]+>)?) (?<argument>[_a-zA-Z0-9]+)(?<after>,|\\))", "${argumentAbstractType}&${argument}${after}", 0),
    // ICounter<int, int> c1;
    // ICounter<int, int>* c1;
    (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^\r\n]+>)?) (?<variable>[_a-zA-Z0-9]+)(?<after> = null)?");), "${abstractType}*${variable}${after}", 0),
    // (expression)
    // expression
    (new Regex(@"\(( | )\((( [a-zA-Z0-9_\*:]+ )\)( , | ; | \))")", "$1$2$3", 0),
    // (method(expression))
    // method(expression)
    (new Regex(@"(?<firstSeparator>\(( | ))\(((?<method>[a-zA-Z0-9_->*:]+)\)((?<expression>((?<parenthesis>\(|(?<-parentesis>\)|[a-zA-Z0-9_->*:]*+)(?<parenthesis>(?!))\)|(?<lastSeparator>(,|;|\\)))"))", "${firstSeparator}${method}({expression})${lastSeparator}", 0),
    // .append(".")
    // .append(1, '.');
    (new Regex(@"\.append\\(\"\"([^\"]|\\\"")\"\"\)\"", ".append(1, '$1')", 0),
    // return ref _elements[node];
    // return &_elements[node];
    (new Regex(@"return ref ([_a-zA-Z0-9]+)\([([_a-zA-Z0-9_\*:]+)\]");), "return &$1[$2]", 0),
    // ((1, 2))
    // ({1, 2})
    (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
    //
```

```

657 (new Regex(@"(?<before>\r?\n[~""\r\n]*("""\|~""\r\n))*""[~""\r\n]*)(?<=\W)new\
    ↪ s+)", "${before}",
    ↪ 10),
658 // x == null
659 // x == nullptr
660 (new Regex(@"(?<before>\r?\n[~""\r\n]*("""\|~""\r\n))*""[~""\r\n]*)(?<=\W)(?<v
    ↪ ariable>[_a-zA-Z][_a-zA-Z0-9]+)(?<operator>\s*(==|!=)\s*)null(?<after>\W)",
    ↪ "${before}${variable}${operator}nullptr${after}", 10),
661 // null
662 // {}
663 (new Regex(@"(?<before>\r?\n[~""\r\n]*("""\|~""\r\n))*""[~""\r\n]*)(?<=\W)null
    ↪ (?<after>\W)", "${before}{}${after}",
    ↪ 10),
664 // default
665 // 0
666 (new Regex(@"(?<before>\r?\n[~""\r\n]*("""\|~""\r\n))*""[~""\r\n]*)(?<=\W)defa
    ↪ ult(?<after>\W)", "${before}0${after}",
    ↪ 10),
667 // object x
668 // void *x
669 (new Regex(@"(?<before>\r?\n[~""\r\n]*("""\|~""\r\n))*""[~""\r\n]*)(?<=\W)(?!
    ↪ @)(object|System\.Object) (?<after>\w)", "${before}void *${after}",
    ↪ 10),
670 // <object>
671 // <void*>
672 (new Regex(@"(?<before>\r?\n[~""\r\n]*("""\|~""\r\n))*""[~""\r\n]*)(?<=\W)(?!
    ↪ @)(object|System\.Object) (?<after>\W)", "${before}void*${after}",
    ↪ 10),
673 // @object
674 // object
675 (new Regex(@"@([_a-zA-Z0-9]+)", "$1", 0),
676 // this->GetType().Name
677 // typeid(this).name()
678 (new Regex(@"(this->GetType\(\)\.Name)", "typeid($1).name()", 0),
679 // ArgumentException
680 // std::invalid_argument
681 (new Regex(@"(?<before>\r?\n[~""\r\n]*("""\|~""\r\n))*""[~""\r\n]*)(?<=\W)(Sys
    ↪ tem\.)?ArgumentNullException(?<after>\W)",
    ↪ "${before}std::invalid_argument${after}", 10),
682 // InvalidOperationException
683 // std::runtime_error
684 (new Regex(@"(\W)(InvalidOperationException|Exception)(\W)",
    ↪ "$1std::runtime_error$3", 0),
685 // ArgumentException
686 // std::invalid_argument
687 (new Regex(@"(\W)(ArgumentException|ArgumentOutOfRangeException)(\W)",
    ↪ "$1std::invalid_argument$3", 0),
688 // template <typename T> struct Range : IEquatable<Range<T>>
689 // template <typename T> struct Range {
690 (new Regex(@"(?<before>template <typename (?<typeParameter>[^\n]+)> (struct|class)
    ↪ (?<type>[_a-zA-Z0-9]+<[^\n]+>)) : (public
    ↪ )?IEquatable<\k<type>>(?<after>(\s|\n)*{)\"", "${before}${after}", 0),
691 // public: delegate void Disposal(bool manual, bool wasDisposed);
692 // public: delegate void Disposal(bool, bool);
693 (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),
694 // public: delegate void Disposal(bool, bool);
695 // using Disposal = void(bool, bool);
696 (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),
697 // <4-1>
698 // <3>
699 (new Regex(@"(?<before><)4-1(?<after>>)", "${before}3${after}", 0),
700 // <3-1>
701 // <2>
702 (new Regex(@"(?<before><)3-1(?<after>>)", "${before}2${after}", 0),
703 // <2-1>
704 // <1>

```

```

705         (new Regex(@"(?<before><)2-1(?<after>>)", "${before}1${after}", 0),
706         // <1-1>
707         // <0>
708         (new Regex(@"(?<before><)1-1(?<after>>)", "${before}0${after}", 0),
709         // #region Always
710         //
711         (new Regex(@"(^\r?\n)[ \t]*#(region|endregion)[^\r\n]*(\r?\n|$)", "", 0),
712         // //define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
713         //
714         (new Regex(@"\\\/[ \t]*#define[ \t]+[_a-zA-Z0-9]+[ \t]*"), "", 0),
715         // #if USEARRAYPOOL\r\n#endif
716         //
717         (new Regex(@"#if [a-zA-Z0-9]+\s+endif", "", 0),
718         // [Fact]
719         //
720         (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),
721         // \A \n ... namespace
722         // \Anamespace
723         (new Regex(@"(\A)(\r?\n)+namespace", "$1namespace", 0),
724         // \A \n ... class
725         // \Aclass
726         (new Regex(@"(\A)(\r?\n)+class", "$1class", 0),
727         // \n\n\n
728         // \n\n
729         (new Regex(@"\r?\n[ \t]*\r?\n[ \t]*\r?\n"), Environment.NewLine +
→      Environment.NewLine, 50),
730         // {\n\n
731         // {\n
732         (new Regex(@"{[ \t]*\r?\n[ \t]*\r?\n"), "{" + Environment.NewLine, 10),
733         // \n\n}
734         // \n}
735         (new Regex(@"\r?\n[ \t]*\r?\n(?<end>[ \t]*)"), Environment.NewLine + "${end}", 10),
736     }.Cast<ISubstitutionRule>().ToList();
737
738     public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
→      base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
739
740     public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
741 }
742 }

```

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

```

1  using Xunit;
2
3  namespace Platform.RegularExpressions.Transformer.CSharpToCpp.Tests
4  {
5      public class CSharpToCppTransformerTests
6      {
7          [Fact]
8          public void EmptyLineTest()
9          {
10             // This test can help to test basic problems with regular expressions like incorrect
→          syntax
11             var transformer = new CSharpToCppTransformer();
12             var actualResult = transformer.Transform("");
13             Assert.Equal("", actualResult);
14         }
15
16         [Fact]
17         public void HelloWorldTest()
18         {
19             const string helloWorldCode = @"using System;
20 class Program
21 {
22     public static void Main(string[] args)
23     {
24         Console.WriteLine("Hello, world!");
25     }
26 }";
27             const string expectedResult = @"class Program
28 {
29     public: static void Main(std::string args[])
30     {
31         printf("Hello, world!\n");
32     }

```

```
33     };";
34
35     var transformer = new CSharpToCppTransformer();
36     var actualResult = transformer.Transform(helloWorldCode);
37     Assert.Equal(expectedResult, actualResult);
38 }
39 }
```

Index

./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs, 17

./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1