

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}${after}", 10),
66 // public ...
67 // public: ...
68 (new Regex(@"(?<newLineAndIndent>r?\n?[
    ↳ \t]*) (?<before>[^\{\\(\r\n)*) (?<access>private|protected|public) [ \t]+(?![^\{\\(\r
    ↳ \n]*((?<=\\s)|\\W) (interface|class|struct) (\\W) [^\{\\(\r\n)*[^\{\\(\r\n)]")",
    ↳ "${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);

```

```

94 // /*~extensionMethod~BuildExceptionString~*/...BuildExceptionString(sb,
95   ↳ exception.InnerException, level + 1);
96 (new Regex(@"(?<before>/\/*~extensionMethod~(?<name>[a-zA-Z0-9]+)~\*/(.\|\\n)+\\W)(?<var_
97   ↳ iable>[_a-zA-Z0-9]+)\\.\\k<name>\\("), "${before}${name}${{variable}}, ",
98   ↳ 50),
99 // Remove markers
100 // /*~extensionMethod~BuildExceptionString~*/
101 //
102 (new Regex(@"\/*~extensionMethod~[a-zA-Z0-9]+~\*/"), "", 0),
103 // (this
104 // (
105 (new Regex(@"\((this ", "(", 0),
106 // private: static readonly Disposal _emptyDelegate = (manual, wasDisposed) => { };
107 // private: inline static std::function<Disposal> _emptyDelegate = [](auto manual,
108   ↳ auto wasDisposed) { };
109 (new Regex(@"(?<access>(private|protected|public): )?static readonly
110   ↳ (?<type>[a-zA-Z][a-zA-Z0-9]*) (?<name>[a-zA-Z_][a-zA-Z0-9_]*) =
111   ↳ \((?<firstArgument>[a-zA-Z_][a-zA-Z0-9_]*)
112   ↳ (?<secondArgument>[a-zA-Z_][a-zA-Z0-9_]*)\)) => {\s*};"), "${access}inline static
113   ↳ std::function<${type}> ${name} = [](auto ${firstArgument}, auto
114   ↳ ${secondArgument}) { };", 0),
115 // public: static readonly EnsureAlwaysExtensionRoot Always = new
116   ↳ EnsureAlwaysExtensionRoot();
117 // public: inline static EnsureAlwaysExtensionRoot Always;
118 (new Regex(@"(?<access>(private|protected|public): )?static readonly
119   ↳ (?<type>[a-zA-Z0-9]+(<[a-zA-Z0-9]+>?)) (?<name>[a-zA-Z0-9_]+) = new
120   ↳ \\k<type>\\(\\);"), "${access}inline static ${type} ${name};", 0),
121 // public: static readonly Range<int> SByte = new
122   ↳ Range<int>(std::numeric_limits<int>::min(), std::numeric_limits<int>::max());
123 // public: inline static Range<int> SByte =
124   ↳ Range<int>(std::numeric_limits<int>::min(), std::numeric_limits<int>::max());
125 (new Regex(@"(?<access>(private|protected|public): )?static readonly
126   ↳ (?<type>[a-zA-Z0-9]+(<[a-zA-Z0-9]+>?)) (?<name>[a-zA-Z0-9_]+) = new
127   ↳ \\k<type>\\((?<arguments>[^\n]+)\\);"), "${access}inline static ${type} ${name} =
128   ↳ ${type}${{arguments}};", 0),
129 // public: static readonly string ExceptionContentsSeparator = "---";
130 // public: inline static std::string ExceptionContentsSeparator = "---";
131 (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly) string
132   ↳ (?<name>[a-zA-Z0-9_]+) = ""(?<string>(\\\\"|\\r\\n)+)"";"), "${access}inline
133   ↳ static std::string ${name} = \\\"${string}\\\";", 0),
134 // private: const int MaxPath = 92;
135 // private: inline static const int MaxPath = 92;
136 (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly)
137   ↳ (?<type>[a-zA-Z0-9]+) (?<name>[_a-zA-Z0-9]+) = (?<value>[^\r\n]+);"),
138   ↳ "${access}inline static const ${type} ${name} = ${value};", 0),
139 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
140   ↳ TArgument : class
141 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
142 (new Regex(@"(?<before> [a-zA-Z]+\\((([a-zA-Z *,]+, |)) (?<type>[a-zA-Z]+) (?<after>(\\
143   ↳ [a-zA-Z *,]+)\\)) [ \\r\\n]+where \\k<type> : class"), "${before}${type}*${after}",
144   ↳ 0),
145 // protected: abstract TElement GetFirst();
146 // protected: virtual TElement GetFirst() = 0;
147 (new Regex(@"(?<access>(private|protected|public): )?abstract
148   ↳ (?<method>[^\r\n]+);"), "${access}virtual ${method} = 0;", 0),
149 // TElement GetFirst();
150 // virtual TElement GetFirst() = 0;
151 (new Regex(@"(?<before>[\\r\\n]+ [ ]+)(?<methodDeclaration>(?!return) [a-zA-Z0-9]+
152   ↳ [a-zA-Z0-9]+\\(([^\\r\\n]*\\)) (?<after>; [ ]*[\\r\\n]+)"), "${before}virtual
153   ↳ ${methodDeclaration} = 0${after}", 1),
154 // protected: readonly TreeElement[] _elements;
155 // protected: TreeElement _elements[N];
156 (new Regex(@"(?<access>(private|protected|public): )?readonly
157   ↳ (?<type>[a-zA-Z<>0-9]+) (\\[\\]]+ ) (?<name>[_a-zA-Z0-9]+);"), "${access}${type}
158   ↳ ${name}[N];", 0),
159 // protected: readonly TElement Zero;
160 // protected: TElement Zero;
161 (new Regex(@"(?<access>(private|protected|public): )?readonly
162   ↳ (?<type>[a-zA-Z<>0-9]+) (?<name>[_a-zA-Z0-9]+);"), "${access}${type} ${name};",
163   ↳ 0),
164 // internal
165 //
166 (new Regex(@"(\\W)internal\\s+"), "$1", 0),
167 // static void NotImplementedException(ThrowExtensionRoot root) => throw new
168   ↳ 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]+)\\(\\(\\(\\^\\)\\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),

```

```

178 // Action<TElement> free
179 // std::function<void(TElement)> free
180 (new Regex(@"Action(<(?<typeParameters>[a-zA-Z0-9]+(,
    ↳ ([a-zA-Z0-9]+))*>)?(?<after>>| (?<variable>[a-zA-Z0-9]+))"),
    ↳ "std::function<void(${typeParameters})>${after}", 0),
181 // Predicate<TArgument> predicate
182 // std::function<bool(TArgument)> predicate
183 (new Regex(@"Predicate(<[a-zA-Z0-9]+> ([a-zA-Z0-9]+)"), "std::function<bool($1)>
    ↳ $2", 0),
184 // var
185 // auto
186 (new Regex(@"(\\W)var(\\W)"), "$1auto$2", 0),
187 // unchecked
188 //
189 (new Regex(@"[\\r\\n]{2}\\s*?unchecked\\s*?$"), "", 0),
190 // throw new
191 // throw
192 (new Regex(@"(\\W)throw new(\\W)"), "$1throw$2", 0),
193 // void RaiseExceptionIgnoredEvent(Exception exception)
194 // void RaiseExceptionIgnoredEvent(const std::exception& exception)
195 (new Regex(@"(\\(| ) (System\\.Exception|Exception) (|\\))"), "$1const
    ↳ std::exception&$3", 0),
196 // EventHandler<Exception>
197 // EventHandler<std::exception>
198 (new Regex(@"(\\W) (System\\.Exception|Exception) (\\W)"), "$1std::exception$3", 0),
199 // override void PrintNode(TElement node, StringBuilder sb, int level)
200 // void PrintNode(TElement node, StringBuilder sb, int level) override
201 (new Regex(@"override ([a-zA-Z0-9 \\*\\+]+) (\\([\\^\\r\\n]+?\\))"), "$1$2 override", 0),
202 // return (range.Minimum, range.Maximum)
203 // return {range.Minimum, range.Maximum}
204 (new Regex(@"(?<before>return\\s*)\\((?<values>[\\^\\n]+)\\) (?!\\() (?<after>\\W)",
    ↳ "${before}${values}${after}", 0),
205 // string
206 // std::string
207 (new Regex(@"(?<before>\\W) (?<!:) string (?<after>\\W)",
    ↳ "${before}std::string${after}", 0),
208 // System.ValueTuple
209 // std::tuple
210 (new Regex(@"(?<before>\\W) (System\\.)?ValueTuple(?!\\s*=|\\() (?<after>\\W)",
    ↳ "${before}std::tuple${after}", 0),
211 // sbyte
212 // std::int8_t
213 (new Regex(@"(?<before>\\W) ((System\\.)?SB|sb)yte(?!\\s*=|\\() (?<after>\\W)",
    ↳ "${before}std::int8_t${after}", 0),
214 // short
215 // std::int16_t
216 (new Regex(@"(?<before>\\W) ((System\\.)?Int16|short) (?!\\s*=|\\() (?<after>\\W)",
    ↳ "${before}std::int16_t${after}", 0),
217 // int
218 // std::int32_t
219 (new Regex(@"(?<before>\\W) ((System\\.)?I|i)nt(32)?(?!\\s*=|\\() (?<after>\\W)",
    ↳ "${before}std::int32_t${after}", 0),
220 // long
221 // std::int64_t
222 (new Regex(@"(?<before>\\W) ((System\\.)?Int64|long) (?!\\s*=|\\() (?<after>\\W)",
    ↳ "${before}std::int64_t${after}", 0),
223 // byte
224 // std::uint8_t
225 (new Regex(@"(?<before>\\W) ((System\\.)?Byte|byte) (?!\\s*=|\\() (?<after>\\W)",
    ↳ "${before}std::uint8_t${after}", 0),
226 // ushort
227 // std::uint16_t
228 (new Regex(@"(?<before>\\W) ((System\\.)?UInt16|ushort) (?!\\s*=|\\() (?<after>\\W)",
    ↳ "${before}std::uint16_t${after}", 0),
229 // uint
230 // std::uint32_t
231 (new Regex(@"(?<before>\\W) ((System\\.)?UI|ui)nt(32)?(?!\\s*=|\\() (?<after>\\W)",
    ↳ "${before}std::uint32_t${after}", 0),
232 // ulong
233 // std::uint64_t
234 (new Regex(@"(?<before>\\W) ((System\\.)?UInt64|ulong) (?!\\s*=|\\() (?<after>\\W)",
    ↳ "${before}std::uint64_t${after}", 0),
235 // char*[] args
236 // char* args[]
237 (new Regex(@"([_a-zA-Z0-9:\\*]?)[\\[] ([a-zA-Z0-9]+)"), "$1 $2[]", 0),
238 // 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()${after}",
    ↳  0),
241 // double.MaxValue
242 // std::numeric_limits<float>::max()
243 (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+\float|double)\.MaxValue(?<after>\W|
    ↳  )"), "${before}std::numeric_limits<${type}>::max()${after}",
    ↳  0),
244 // using Platform.Numbers;
245 //
246 (new Regex(@"([\r\n]{2}|~)\s*?using [\a-zA-Z0-9]+\s*?$"), "", 0),
247 // struct TreeElement { }
248 // struct TreeElement { };
249 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([~;])", "$1
    ↳  $2$3{$4};$5", 0),
250 // class Program { }
251 // class Program { };
252 (new Regex(@"(?<type>struct|class)
    ↳  (?<name>[a-zA-Z0-9]+[~\r\n]*) (?<beforeBody>[\r\n]+(?<indentLevel>[\t
    ↳  ]*)?)\{(?<body>[\S\s]+?[~\r\n]+\k<indentLevel>)\}(?<afterBody>[~;]|$)", "${type}
    ↳  ${name}${beforeBody}${body}};${afterBody}", 0),
253 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
254 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
255 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(<[a-zA-Z0-9 ,]+>)? : ([a-zA-Z0-9]+)",
    ↳  "$1 $2$3 : public $4", 0),
256 // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
257 // class IProperty : public ISetter<TValue, TObject>, public IProvider<TValue,
    ↳  TObject>
258 (new Regex(@"(?<before>(struct|class) [a-zA-Z0-9]+ : ((public
    ↳  [a-zA-Z0-9]+(<[a-zA-Z0-9 ,]+>)?,
    ↳  )+)?(?<inheritedType>(?!public)[a-zA-Z0-9]+(<[a-zA-Z0-9 ,]+>)?)(?<after>(,
    ↳  [a-zA-Z0-9]+(?!>)|[\r\n]+))", "${before}public ${inheritedType}${after}", 10),
259 // Insert scope borders.
260 // ref TElement root
261 // ~!root!~ref TElement root
262 (new Regex(@"(?<definition>(?!<= |\) (ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<ref>))
    ↳  (?<variable>[a-zA-Z0-9]+)(?<= | | =))", "~!${variable}!~!${definition}", 0),
263 // Inside the scope of ~!root!~ replace:
264 // root
265 // *root
266 (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),
267 // Remove scope borders.
268 // ~!root!~
269 //
270 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", 5),
271 // ref auto root = ref
272 // ref auto root =
273 (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)", "$1* $2 =$3", 0),
274 // *root = ref left;
275 // root = left;
276 (new Regex(@"*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)", "$1 = $2$3", 0),
277 // (ref left)
278 // (left)
279 (new Regex(@"\ (ref ([a-zA-Z0-9]+) (\) | \ (| ,) )", "($1$2", 0),
280 // ref TElement
281 // TElement*
282 (new Regex(@"( | \() ref ([a-zA-Z0-9]+) ", "$1$2* ", 0),
283 // ref sizeBalancedTree.Root
284 // &sizeBalancedTree->Root
285 (new Regex(@"ref ([a-zA-Z0-9]+) \. ([a-zA-Z0-9\*]+)", "&$1->$2", 0),
286 // ref GetElement(node).Right
287 // &GetElement(node)->Right
288 (new Regex(@"ref ([a-zA-Z0-9]+) \ ( ([a-zA-Z0-9\*]+) \) \. ([a-zA-Z0-9]+) ",
    ↳  "&$1($2)->$3", 0),
289 // GetElement(node).Right
290 // GetElement(node)->Right
291 (new Regex(@"([a-zA-Z0-9]+) \ ( ([a-zA-Z0-9\*]+) \) \. ([a-zA-Z0-9]+) ", "$1($2)->$3", 0),
292 // [Fact] \npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
293 // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
294 (new Regex(@"\[Fact\] [\s\n]+(public: )?(static )?void ([a-zA-Z0-9]+) \ ( \) )", "public:
    ↳  TEST_METHOD($3)", 0),
295 // 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(\$\"${left}\").append(Platform::Converters::To<std::string>(${expres
→ sion})) .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\\((Platf
→ orm::Converters::To<std::string>\\([\^]\n)+\\)|[\^]\n)+))\\).append\"",
→ "${begin}.append", 10),
// Console.WriteLine("...")
// printf("...\n")
(new Regex(@"Console\.WriteLine\"(\"([\\\"\\\r\n]+)\"\"\"\""), "printf(\"$1\\n\\n\"", 0),
// TElement Root;
// TEElement Root = 0;
(new Regex(@"(?:<before>\r?\n[\\t ]+)(?:<access>(private|protected|public)(:
→ )?)?(?:<type>[a-zA-Z0-9:_]+(?:<!return>) (?:<name>[_a-zA-Z0-9]+);)",
→ "${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 TEelement[MaxPath];
// TEelement 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
345   static std::vector<std::exception> _exceptionsBag;
346 (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),
347 // public: static IReadonlyCollection<std::exception> GetCollectedExceptions() {
   return _exceptionsBag; }
348 // public: static std::vector<std::exception> GetCollectedExceptions() { return
   std::vector<std::exception>(_exceptionsBag); }
349 (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}>(${fieldName}); }", 0),
350 // public: static event EventHandler<std::exception> ExceptionIgnored =
   OnExceptionIgnored; ... };
351 // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
   const std::exception&> ExceptionIgnored = OnExceptionIgnored; };
352 (new Regex(@"(?<begin>\r?\n(\r?\n)?(?<halfIndent>[
   \t]+)\k<halfIndent>)(?<access>(private|protected|public): )?static event
   EventHandler<(?<argumentType>[~;\r\n]+)> (?<name>[_a-zA-Z0-9]+) = (?<defaultDele
   gate>[_a-zA-Z0-9]+);(?<middle>(.|\n)+)?(?<end>\r?\n\k<halfIndent>);)"),
   "${middle}" + Environment.NewLine + Environment.NewLine +
   "${halfIndent}${halfIndent}${access}static inline
   Platform::Delegates::MulticastDelegate<void(void*, const ${argumentType}&)>
   ${name} = ${defaultDelegate};${end}", 0),
353 // public: event Disposal OnDispose;
354 // public: Platform::Delegates::MulticastDelegate<Disposal> OnDispose;
355 (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}> ${name};", 0),
356 // Insert scope borders.
357 // class IgnoredExceptions { ... private: inline static std::vector<std::exception>
   _exceptionsBag;
358 // class IgnoredExceptions { /*~_exceptionsBag~*/ ... private: inline static
   std::vector<std::exception> _exceptionsBag;
359 (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[ \t ]*)class [~{\r\n}+\r\n[ \t
   ]*{ }(?<middle>((?!class)\.|\n)+)?(?<vectorFieldDeclaration>(?!<access>(private|pro
   tected|public): )inline static std::vector<(?<argumentType>[~;\r\n]+)>
   (?<fieldName>[_a-zA-Z0-9]+);)"),
   "${classDeclarationBegin}/*~${fieldName}~*/${middle}${vectorFieldDeclaration}",
   0),
360 // Inside the scope of ~!_exceptionsBag!~ replace:
361 // _exceptionsBag.Add(exception);
362 // _exceptionsBag.push_back(exception);
363 (new Regex(@"(?<scope>\/\~(?<fieldName>[_a-zA-Z0-9]+)\~\*/)(?<separator>.\|\n)(?<befor
   e>((?!\/\~\~\k<fieldName>\~\*/)(.|\n))*?)\k<fieldName>\.Add"),
   "${scope}${separator}${before}${fieldName}.push_back", 10),
364 // Remove scope borders.
365 // /*~_exceptionsBag~*/
366 //
367 (new Regex(@"\/\~[_a-zA-Z0-9]+\~\*/"), "", 0),
368 // Insert scope borders.
369 // class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
370 // class IgnoredExceptions { /*~_exceptionsBag~*/ ... private: static std::mutex
   _exceptionsBag_mutex;
371 (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[ \t ]*)class [~{\r\n}+\r\n[ \t
   ]*{ }(?<middle>((?!class)\.|\n)+)?(?<mutexDeclaration>private: inline static
   std::mutex (?<fieldName>[_a-zA-Z0-9]+)_mutex;)"),
   "${classDeclarationBegin}/*~${fieldName}~*/${middle}${mutexDeclaration}", 0),
372 // Inside the scope of ~!_exceptionsBag!~ replace:
373 // return std::vector<std::exception>(_exceptionsBag);
374 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
   std::vector<std::exception>(_exceptionsBag);
375 (new Regex(@"(?<scope>\/\~(?<fieldName>[_a-zA-Z0-9]+)\~\*/)(?<separator>.\|\n)(?<befor
   e>((?!\/\~\~\k<fieldName>\~\*/)(.|\n))*?)\{(?<after>((?!lock_guard)[~{};\r\n])*\k<f
   ieldName>[~;}\r\n]*;)", "${scope}${separator}${before}{
   std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", 10),
376 // Inside the scope of ~!_exceptionsBag!~ replace:

```



```

376 // _exceptionsBag.Add(exception);
377 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
378 → _exceptionsBag.Add(exception);
379 (new Regex(@"(?<scope>/\~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
380 → e((?!/\~\k<fieldName>~\*/)(.\|\\n))*?){(?<after>((?!lock_guard)([~{};]|\\n))*?\r\n
381 → ?\\n(?<indent>[ \t]*)\k<fieldName>[~;]\r\n]*;)" ),
382 → "${scope}${separator}${before}{ " + Environment.NewLine +
383 → "${indent}std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", 10),
384 // Remove scope borders.
385 // /*~_exceptionsBag~*/
386 //
387 (new Regex(@"/*~[_a-zA-Z0-9]+~\*/"), "", 0),
388 // Insert scope borders.
389 // class IgnoredExceptions { ... public: static inline
390 → Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
391 → ExceptionIgnored = OnExceptionIgnored;
392 // class IgnoredExceptions { /*~ExceptionIgnored~*/ ... public: static inline
393 → Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
394 → ExceptionIgnored = OnExceptionIgnored;
395 (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[ \t ]*)class [~{r\\n}]+\r\n[ \t
396 → ]*)(?<middle>((?!class).\|\\n)+?)(?<eventDeclaration>(?<access>(private|protected|
397 → |public): )static inline
398 → Platform::Delegates::MulticastDelegate<(?<argumentType>[~;\r\\n]+)>
399 → (?<name>[_a-zA-Z0-9]+) = (?<defaultDelegate>[_a-zA-Z0-9]+);)" ),
400 → "${classDeclarationBegin}/*~${name}~*/${middle}${eventDeclaration}", 0),
401 // Inside the scope of ~!ExceptionIgnored!~ replace:
402 // ExceptionIgnored.Invoke(NULL, exception);
403 // ExceptionIgnored(NULL, exception);
404 (new Regex(@"(?<scope>/\~(?<eventName>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
405 → >((?!/\~\k<eventName>~\*/)(.\|\\n))*?)\k<eventName>\\.Invoke)" ),
406 → "${scope}${separator}${before}${eventName}", 10),
407 // Remove scope borders.
408 // /*~ExceptionIgnored~*/
409 //
410 (new Regex(@"/*~[_a-zA-Z0-9]+~\*/"), "", 0),
411 // Insert scope borders.
412 // auto added = new StringBuilder();
413 // /*~sb~*/std::string added;
414 (new Regex(@"(auto|(System\\.Text\\.)?StringBuilder) (?<variable>[_a-zA-Z0-9]+) = new
415 → (System\\.Text\\.)?StringBuilder\\(\\);)", "/*~${variable}~*/std::string
416 → ${variable};", 0),
417 // static void Indent(StringBuilder sb, int level)
418 // static void Indent(/*~sb~*/StringBuilder sb, int level)
419 (new Regex(@"(?<start>, |\\() (System\\.Text\\.)?StringBuilder
420 → (?<variable>[_a-zA-Z0-9]+)(?<end>, |\\))", "${start}/*~${variable}~*/std::string&
421 → ${variable}${end}", 0),
422 // Inside the scope of ~!added!~ replace:
423 // sb.ToString()
424 // sb
425 (new Regex(@"(?<scope>/\~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
426 → ((?!/\~\k<variable>~\*/)(.\|\\n))*?)\k<variable>\\.ToString\\(\\)",
427 → "${scope}${separator}${before}${variable}", 10),
428 // sb.AppendLine(argument)
429 // sb.append(Platform::Converters::To<std::string>(argument)).append(1, '\\n')
430 (new Regex(@"(?<scope>/\~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
431 → ((?!/\~\k<variable>~\*/)(.\|\\n))*?)\k<variable>\\.AppendLine\\((?<argument>[~\\n]+)\\)",
432 → "${scope}${separator}${before}${variable}.append(Platform::Converters::To<std::s
433 → tring>(${argument})).append(1, '\\n')",
434 → 10),
435 // sb.Append('\\t', level);
436 // sb.append(level, '\\t');
437 (new Regex(@"(?<scope>/\~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
438 → ((?!/\~\k<variable>~\*/)(.\|\\n))*?)\k<variable>\\.Append\\('(?<character>[~'\\r\\n]
439 → +)', (?<count>[~\\n]+)\\)",
440 → "${scope}${separator}${before}${variable}.append(${count}, '${character}')" , 10),
441 // sb.Append(argument)
442 // sb.append(Platform::Converters::To<std::string>(argument))
443 (new Regex(@"(?<scope>/\~(?<variable>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
444 → ((?!/\~\k<variable>~\*/)(.\|\\n))*?)\k<variable>\\.Append\\((?<argument>[~\\n]+)\\)",
445 → "${scope}${separator}${before}${variable}.append(Platform::Converters::To<std::s
446 → tring>(${argument}))",
447 → 10),
448 // Remove scope borders.
449 // /*~sb~*/

```

```

417 //
418 (new Regex(@"/*~[a-zA-Z0-9]+~*/"), "", 0),
419 // Insert scope borders.
420 // auto added = new HashSet<TElement>();
421 // ~!added!~std::unordered_set<TElement> added;
422 (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
    ↳ HashSet<(?<element>[a-zA-Z0-9]+)>\(\(\);"),
    ↳ "~!${variable}!~std::unordered_set<${element}> ${variable};", 0),
423 // Inside the scope of ~!added!~ replace:
424 // added.Add(node)
425 // added.insert(node)
426 (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),
427 // Inside the scope of ~!added!~ replace:
428 // added.Remove(node)
429 // added.erase(node)
430 (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(${argument})", 10),
431 // if (added.insert(node)) {
432 // if (!added.contains(node)) { added.insert(node);
433 (new Regex(@"if \\((?<variable>[a-zA-Z0-9]+)\\.insert\\((?<argument>[a-zA-Z0-9]+)\\)\\)(?
    ↳ <separator>[\\t ]*[\\r\\n]+)(?<indent>[\\t ]*){", "if
    ↳ (!${variable}.contains(${argument})) ${separator}${indent}{ " +
    ↳ Environment.NewLine + "${indent} ${variable}.insert(${argument});", 0),
434 // Remove scope borders.
435 // ~!added!~
436 //
437 (new Regex(@"~![a-zA-Z0-9]+!~"), "", 5),
438 // Insert scope borders.
439 // auto random = new System.Random(0);
440 // std::srand(0);
441 (new Regex(@"[a-zA-Z0-9\\.]+ ([a-zA-Z0-9]+) = new
    ↳ (System\\.)?Random\\((?<[a-zA-Z0-9]+)\\)", "~!$1!~std::srand($3);", 0),
442 // Inside the scope of ~!random!~ replace:
443 // random.Next(1, N)
444 // (std::rand() % N) + 1
445 (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.\|\\n)(?<before>((?<
    ↳ !~!\k<variable>!~)(.\|\\n)*?)\k<variable>\.Next\\((?<from>[a-zA-Z0-9]+),
    ↳ (?<to>[a-zA-Z0-9]+)\\)", "${scope}${separator}${before}(std::rand() % ${to}) +
    ↳ ${from}", 10),
446 // Remove scope borders.
447 // ~!random!~
448 //
449 (new Regex(@"~![a-zA-Z0-9]+!~"), "", 5),
450 // Insert method body scope starts.
451 // void PrintNodes(TElement node, StringBuilder sb, int level) {
452 // void PrintNodes(TElement node, StringBuilder sb, int level) { /*method-start*/
453 (new Regex(@"(?<start>\\r?\\n[\\t ]+)(?<prefix>((private|protected|public): )?(virtual
    ↳ )?[a-zA-Z0-9:_]+
    ↳ )?(?<method>[a-zA-Z][a-zA-Z0-9]*)\\((?<arguments>[^\n]*)\\)(?<override>(
    ↳ override)?)(?<separator>[\\t\\r\\n]*)\\{(?<end>[^\n])", "${start}${prefix}${method}
    ↳ (${arguments})${override}${separator}{ /*method-start*/${end}",
    ↳ 0),
454 // Insert method body scope ends.
455 // { /*method-start*/...}
456 // { /*method-start*/... /*method-end*/}
457 (new Regex(@"{ /*method-start*/(?<body>((?<bracket>\\{) | (?<-bracket>\\}) | [^\n\{\}]*)+
    ↳ \\})", "{ /*method-start*/${body} /*method-end*/}",
    ↳ 0),
458 // Inside method bodies replace:
459 // GetFirst(
460 // this->GetFirst(
461 (new
    ↳ Regex(@"(?<scope>/\\*method-start\\*/)(?<before>((?!/\\*method-end\\*/)(.\|\\n))*?) (?
    ↳ <separator>[\\W] (?<!(?:\\.|->|throw\\s+)))(?<method>(?!sizeof) [a-zA-Z0-9]+)\\((?!\\
    ↳ \\{) (?<after>(.\\|\\n)*?) (?<scopeEnd>/\\*method-end\\*/)",
    ↳ "${scope}${before}${separator}this->${method} (${after}${scopeEnd}", 100),
462 // Remove scope borders.
463 // /*method-start*/
464 //
465 (new Regex(@"/\\*method-(start|end)\\*/"), "", 0),
466 // Insert scope borders.
467 // const std::exception& ex
468 // 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 //

```

```

507 (new Regex(@"/*~[_a-zA-Z0-9]+~\*/", "", 0),
508 // Insert scope borders.
509 // std::tuple<T, T> tuple
510 // std::tuple<T, T> tuple/*~tuple~/
511 (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?tuple<[^\n]+>&?
    → (?<variable>[_a-zA-Z0-9]+))(?<after>\W)"),
    → "${before}${variableDefinition}/*~${variable}*/${after}", 0),
512 // Inside the scope of ~!ex!~ replace:
513 // tuple.Item1
514 // std::get<1-1>(tuple)
515 (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),
516 // Remove scope borders.
517 // /*~ex~/
518 //
519 (new Regex(@"/*~[_a-zA-Z0-9]+~\*/", "", 0),
520 // Insert scope borders.
521 // class Range<T> {
522 // class Range<T> {/\*~type~Range<T>~/
523 (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),
524 // Inside the scope of /*~type~Range<T>~/ insert inner scope and replace:
525 // public: static implicit operator std::tuple<T, T>(Range<T> range)
526 // public: operator std::tuple<T, T>() const {/\*~variable~Range<T>~/
527 (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),
528 // Inside the scope of /*~type~Range<T>~/ replace:
529 // 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)); }
530 // public: Range(std::tuple<T, T> tuple) : Range(std::get<1-1>(tuple),
    → std::get<2-1>(tuple)) { }
531 (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),
532 // Inside the scope of /*~variable~range~/ replace:
533 // range.Minimum
534 // this->Minimum
535 (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}${after}", 10),
536 // Remove scope borders.
537 // /*~ex~/
538 //
539 (new Regex(@"/*~[^\n\*]+~[^\n\*]+~\*/", "", 0),
540 // Insert scope borders.
541 // namespace Platform::Ranges { ... }
542 // namespace Platform::Ranges {/\*~start~namespace~Platform::Ranges~/ ...
    → /\*~end~namespace~Platform::Ranges~/
543 (new Regex(@"(?<namespaceDeclarationBegin>\r?\n(?<indent>[\t ]*)namespace
    → (?<namespaceName>(?!<namePart>[_a-zA-Z] [_a-zA-Z0-9]+)(?<nextNamePart>::[_a-zA-Z] [_a-z
    → A-Z0-9]+)+)(\s|\n)*)(?<middle>.\|\n)*)(?<end>(?!<= \r?\n \k<indent>}{?!;))"),
    → "${namespaceDeclarationBegin}/*~start~namespace~${namespaceName}~/${middle}/*~e
    → nd~namespace~${namespaceName}~/${end}",
    → 0),
544 // Insert scope borders.
545 // class Range<T> { ... };
546 // class Range<T> {/\*~start~type~Range<T>~T~/ ... /\*~end~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>/\s*~start~namespace~(?<namespace>[^\n\*]+)~\s*/)
    (?<betweenStartScopes>(\.|\n)+)(?<typeScopeStart>/\s*~start~type~(?<type>[^\n\*]+)
    )~(?<typeParameter>[^\n\*]+)~\s*/)(?<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>/\s*~end~type~\k<type>~\k<typeParameter>~\s*/)(?<betweenEndSc
    pes>(\.|\n)+)(?<namespaceScopeEnd>/\s*~end~namespace~\k<namespace>~\s*/)\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>/\s*~start~method~\s*/)(?<before>.+({|,
    ))(?<name>[a-zA-Z][a-zA-Z0-9]+)(?<after>[^\n\.\(a-zA-Z0-9)(?!/\s*~end~method~\s*/)
    ][^\n]+)(?<methodScopeEnd>/\s*~end~method~\s*/)",
    @"${methodScopeStart}${before}obj.${name}${after}${methodScopeEnd}", 10),
556 // Remove scope borders.
557 // /*~start~type~Range<T>~*/
558 //
559 (new Regex(@"\/\s*~[^\n\*]+(~[^\n\*]+)*~\s*/"), "", 0),
560 // class Disposable<T> : public Disposable
561 // class Disposable<T> : public Disposable<>
562 (new Regex(@"(?<before>(struct|class) (?<type>[a-zA-Z][a-zA-Z0-9]*)<[^\<>\n]+> :
    (?<access>(private|protected|public) )?\k<type>)(?<after>\b(?:!<))"),
    @"${before}<>${after}", 0),
563 // Insert scope borders.
564 // class Disposable<T> : public Disposable<> { ... };
565 // class Disposable<T> : public Disposable<>
    { /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/ ...
    /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/};
566 (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)template[\t
    ]*(?<typeParameters>[^\n]*)>[\t ]*(struct|class)[\t
    ]+(?<fullType>(?!<type>[a-zA-Z][a-zA-Z0-9]*)(<[^\<>\n]*>)?)[\t ]*:[\t
    ]*(?<access>(private|protected|public)[\t
    ]+)?(?<fullBaseType>(?!<baseType>[a-zA-Z][a-zA-Z0-9]*)(<[^\<>\n]*>)?)[\t
    ]*(\r?\n)?[\t
    ]*(?<middle>(\.|\n)*)(?<beforeEnd>(?!<=\\r?\\n\\k<indent>)(?<end>});)"),
    @"${classDeclarationBegin}/*~start~type~${type}~${fullType}~${baseType}~${fullBas
    eType}~*/${middle}${beforeEnd}/*~end~type~${type}~${fullType}~${baseType}~${full
    BaseType}~*/${end}",
    0),
567 // Inside scopes replace:
568 // /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/ ... ) : base(
    ... /*~end~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/
569 // /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/ ... ) :
    Disposable<>( /*~end~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/

```

```

570 (new Regex(@"(?<before>(?(typeScopeStart>/\~*start~type~(?<types>(?(type>[~\n\*]+)~
    ↳ (?(fullType>[~\n\*]+)~\k<type>~(?(fullBaseType>[~\n\*]+))~\*/)(.|\n)+?)\s*:\s
    ↳ )*base(?(after>\((.|\n)+?(?<typeScopeEnd>/\~*end~type~\k<types>~\*/))"),
    ↳ "${before}${fullBaseType}${after}", 20),
571 // Inside scopes replace:
572 // /\~*start~type~Disposable~Disposable<T>~X~X<>~*/ ... ) : base( ...
    ↳ /\~*end~type~Disposable~Disposable<T>~X~X<>~*/
573 // /\~*start~type~Disposable~Disposable<T>~X~X<>~*/ ... ) : X(
    ↳ /\~*end~type~Disposable~Disposable<T>~X~X<>~*/
574 (new Regex(@"(?<before>(?(typeScopeStart>/\~*start~type~(?<types>(?(type>[~\n\*]+)~
    ↳ (?(fullType>[~\n\*]+)~(?(baseType>[~\n\*]+)~(?(fullBaseType>[~\n\*]+))~\*/)(.|\n)+?)\s*:\s
    ↳ )*base(?(after>\((.|\n)+?(?<typeScopeEnd>/\~*end~type~\k<types>~\*/))"),
    ↳ "${before}${baseType}${after}",
    ↳ 20),
575 // Inside scopes replace:
576 // /\~*start~type~Disposable~Disposable<T>~X~X<>~*/ ... public: Disposable(T object)
    ↳ { Object = object; } ... public: Disposable(T object) : Disposable(object) { }
    ↳ ... /\~*end~type~Disposable~Disposable<T>~X~X<>~*/
577 // /\~*start~type~Disposable~Disposable<T>~X~X<>~*/ ... public: Disposable(T object)
    ↳ { Object = object; } /\~*end~type~Disposable~Disposable<T>~X~X<>~*/
578 (new Regex(@"(?<before>(?(typeScopeStart>/\~*start~type~(?<types>(?(type>[~\n\*]+)~
    ↳ (?(fullType>[~\n\*]+)~(?(baseType>[~\n\*]+)~(?(fullBaseType>[~\n\*]+))~\*/)(.|\n)+?(?<constructor>(?(access>(private|protected|public):[\t
    ↳ ]*)?\k<type>\((?(arguments>[~() \n\+])\)\s*{[~{} \n\+]})(.|\n)+?)*(?<duplicateConstr
    ↳ uctor>(?(access>(private|protected|public):[\t
    ↳ ]*)?\k<type>\((\k<arguments>)\)\s*:[~{} \n\+]\s*{[~{} \n\+]})(?(after>(.|\n)+?(?<typeS
    ↳ copeEnd>/\~*end~type~\k<types>~\*/))"), "${before}${after}",
    ↳ 20),
579 // Remove scope borders.
580 // /\~*start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/
581 //
582 (new Regex(@"\/\~*[~\*\n\+](~[~\*\n\+])*~\*/"), "", 0),
583 }.Cast<ISubstitutionRule>().ToList();
584
585 public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
586 {
587     // ICounter<int, int> c1;
588     // ICounter<int, int>* c1;
589     (new Regex(@"(?<abstractType>I[A-Z] [a-zA-Z0-9]+(<[~>\r\n]+)?))
    ↳ (?(variable>[_a-zA-Z0-9]+)(?(after> = null)?);"), "${abstractType}*
    ↳ ${variable}${after};", 0),
590     // (expression)
591     // expression
592     (new Regex(@"(\(|\)|)(([a-zA-Z0-9_\*:])(,| |;|\))"), "$1$2$3", 0),
593     // (method(expression))
594     // method(expression)
595     (new Regex(@"(?<firstSeparator>(\(|
    ↳ ))\((?<method>[a-zA-Z0-9_\*:])(?<expression>((?<parenthesis>\(|(?<-parent
    ↳ hesis>)|[a-zA-Z0-9_\*:]*)+)(?(parenthesis)(?!))\))\)(?<lastSeparator>(,|
    ↳ |;|\))"), "${firstSeparator}${method}(${expression})${lastSeparator}", 0),
596     // .append(" ")
597     // .append(1, '.');
598     (new Regex(@"\.(append\(""[^\\"]*"|\\[""])*\(")"", ".append(1, '$1')", 0),
599     // return ref _elements[node];
600     // return &elements[node];
601     (new Regex(@"return ref ([_a-zA-Z0-9]+)\([([_a-zA-Z0-9_\*:])(;)"", "return &$1[$2];",
    ↳ 0),
    ↳ // ((1, 2))
    ↳ // ({1, 2})
602     (new Regex(@"(?<before>\(|,)\((?<first>[~\n()]+),
    ↳ (?(second>[~\n()]+)\)(?(after>\(|,)"", "${before}${{first},
    ↳ ${second}}${after}", 10),
    ↳ // {1, 2}.GetHashCode()
    ↳ // Platform::Hashing::Hash(1, 2)
603     (new Regex(@"{?(?<first>[~\n{}]+), (?(second>[~\n{}]+)}\.(GetHashCode\(\))",
    ↳ "Platform::Hashing::Hash(${first}, ${second})", 10),
    ↳ // range.ToString()
    ↳ // Platform::Converters::To<std::string>(range).data()
604     (new Regex(@"(?<before>\W)(?(variable>[_a-zA-Z] [_a-zA-Z0-9]+)\.(ToString\(\))",
    ↳ "${before}Platform::Converters::To<std::string>(${variable}).data()", 10),
    ↳ // new
    ↳ //
605     (new Regex(@"(?<before>\r?\n[""]\r\n)*([""]\r\n|[""]\r\n)*[""]^[""]\r\n)*)(?<=\\W)new\
    ↳ s+)"", "${before}",
    ↳ 10),

```



```

614 // x == null
615 // x == nullptr
616 (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),
617 // null
618 // {}
619 (new Regex(@"(?<before>\r?\n[~""\r\n]*(""(\\"""|["""\r\n])*""["""\r\n]*)*) (?<=\\W)null_
→ (?<after>\\W)"), "${before}{}${after}",
→ 10),
620 // default
621 // 0
622 (new Regex(@"(?<before>\r?\n[~""\r\n]*(""(\\"""|["""\r\n])*""["""\r\n]*)*) (?<=\\W)defa_
→ ult(?<after>\\W)"), "${before}0${after}",
→ 10),
623 // object x
624 // void *x
625 (new Regex(@"(?<before>\r?\n[~""\r\n]*(""(\\"""|["""\r\n])*""["""\r\n]*)*) (?<=\\W) (?<_!_
→ @)(object|System\\.Object) (?<after>\\w)"), "${before}void *${after}",
→ 10),
626 // <object>
627 // <void*>
628 (new Regex(@"(?<before>\r?\n[~""\r\n]*(""(\\"""|["""\r\n])*""["""\r\n]*)*) (?<=\\W) (?<_!_
→ @)(object|System\\.Object) (?<after>\\W)"), "${before}void*${after}",
→ 10),
629 // @object
630 // object
631 (new Regex(@"@(_a-zA-Z0-9)+"), "$1", 0),
632 // this->GetType().Name
633 // typeid(this).name()
634 (new Regex(@"(this)->GetType\\(\\)\\.Name"), "typeid($1).name()", 0),
635 // ArgumentNullException
636 // std::invalid_argument
637 (new Regex(@"(?<before>\r?\n[~""\r\n]*(""(\\"""|["""\r\n])*""["""\r\n]*)*) (?<=\\W) (Sys_
→ tem\\.)?ArgumentNullException(?<after>\\W)"),
→ "${before}std::invalid_argument${after}", 10),
638 // InvalidOperationException
639 // std::runtime_error
640 (new Regex(@"(\\W)(InvalidOperationException|Exception)(\\W)"),
→ "$1std::runtime_error$3", 0),
641 // ArgumentException
642 // std::invalid_argument
643 (new Regex(@"(\\W)(ArgumentException|ArgumentOutOfRangeException)(\\W)"),
→ "$1std::invalid_argument$3", 0),
644 // template <typename T> struct Range : IEquatable<Range<T>>
645 // template <typename T> struct Range {
646 (new Regex(@"(?<before>template <typename (?<typeParameter>[~\n]+) > (struct|class)
→ (?<type>[_a-zA-Z0-9]+<[~\n]+>)) : (public
→ )?IEquatable<\k<type>>(?<after>(\\s|\\n)*{)"), "${before}${after}", 0),
647 // public: delegate void Disposal(bool manual, bool wasDisposed);
648 // public: delegate void Disposal(bool, bool);
649 (new Regex(@"(?<before>(?!<access>(private|protected|public): )delegate
→ (?<returnType>[_a-zA-Z][_a-zA-Z0-9:]+)
→ (?<delegate>[_a-zA-Z][_a-zA-Z0-9:]+)\\(((?!<leftArgumentType>[_a-zA-Z][_a-zA-Z0-9:]+),
→ )*) (?<argumentType>[_a-zA-Z][_a-zA-Z0-9:]+)
→ (?<argumentName>[_a-zA-Z][_a-zA-Z0-9:]+) (?<after>(,
→ (?<rightArgumentType>[_a-zA-Z][_a-zA-Z0-9:]+)
→ (?<rightArgumentName>[_a-zA-Z][_a-zA-Z0-9:]+))*\\);)"),
→ "${before}${argumentType}${after}", 20),
650 // public: delegate void Disposal(bool, bool);
651 // using Disposal = void(bool, bool);
652 (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),
653 // <4-1>
654 // <3>
655 (new Regex(@"(?<before><)4-1(?<after>>)", "${before}3${after}", 0),
656 // <3-1>
657 // <2>
658 (new Regex(@"(?<before><)3-1(?<after>>)", "${before}2${after}", 0),
659 // <2-1>
660 // <1>
661 (new Regex(@"(?<before><)2-1(?<after>>)", "${before}1${after}", 0),
662 // <1-1>
663 // <0>

```

```

664 (new Regex(@"(?<before><)1-1(?<after>>)", "${before}0${after}", 0),
665 // #region Always
666 //
667 (new Regex(@"(^\r?\n)[ \t]*#(region|endregion)[^\r\n]*(\r?\n|$)", "", 0),
668 // // #define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
669 //
670 (new Regex(@"\\\/[ \t]*#define[ \t]+[_a-zA-Z0-9]+[ \t]*"), "", 0),
671 // #if USEARRAYPOOL\r\n#endif
672 //
673 (new Regex(@"#if [_a-zA-Z0-9]+\s+#endif", "", 0),
674 // [Fact]
675 //
676 (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),
677 // \A \n ... namespace
678 // \Anamespace
679 (new Regex(@"(\A)(\r?\n)+namespace", "$1namespace", 0),
680 // \A \n ... class
681 // \Aclass
682 (new Regex(@"(\A)(\r?\n)+class", "$1class", 0),
683 // \n\n\n
684 // \n\n
685 (new Regex(@"\r?\n[ \t]*\r?\n[ \t]*\r?\n"), Environment.NewLine +
→ Environment.NewLine, 50),
686 // {\n\n
687 // {\n
688 (new Regex(@"[ \t]*\r?\n[ \t]*\r?\n"), "{" + Environment.NewLine, 10),
689 // \n\n}
690 // \n}
691 (new Regex(@"\r?\n[ \t]*\r?\n(?<end>[ \t]*)"), Environment.NewLine + "${end}", 10),
692 }.Cast<ISubstitutionRule>().ToList();
693
694 public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
695     → base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
696
697 public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
698 }

```

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
11             → syntax
12             var transformer = new CSharpToCppTransformer();
13             var actualResult = transformer.Transform("");
14             Assert.Equal("", actualResult);
15         }
16
17         [Fact]
18         public void HelloWorldTest()
19         {
20             const string helloWorldCode = @"using System;
21 class Program
22 {
23     public static void Main(string[] args)
24     {
25         Console.WriteLine(""Hello, world!"");
26     }
27 }";
28             const string expectedResult = @"class Program
29 {
30     public: static void Main(std::string args[])
31     {
32         printf(""Hello, world!\n"");
33     }
34 };";
35             var transformer = new CSharpToCppTransformer();
36             var actualResult = transformer.Transform(helloWorldCode);

```

```
36         Assert.Equal(expectedResult, actualResult);
37     }
38 }
39 }
```

Index

- ./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs, 16
- ./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1