

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     /// <summary>
11     /// <para>
12     /// Represents the sharp to cpp transformer.
13     /// </para>
14     /// <para></para>
15     /// </summary>
16     /// <seealso cref="TextTransformer"/>
17     public class CSharpToCppTransformer : TextTransformer
18     {
19         /// <summary>
20         /// <para>
21         /// The to list.
22         /// </para>
23         /// <para></para>
24         /// </summary>
25         public static readonly IList<ISubstitutionRule> FirstStage = new List<SubstitutionRule>
26         {
27             // // ...
28             //
29             (new Regex(@"(\r?\n)?[ \t]+//+.+"), "", 0),
30             // #pragma warning disable CS1591 // Missing XML comment for publicly visible type
31             // or member
32             //
33             (new Regex(@"^~\s*?#pragma[sa-zA-Z0-9]+$"), "", 0),
34             // { \n \n \n
35             // {
36             (new Regex(@"{\s+[\r\n]+") , "{" + Environment.NewLine, 0),
37             // Platform.Collections.Methods.Lists
38             // Platform::Collections::Methods::Lists
39             (new Regex(@"(namespace[\r\n]+?)\.([\r\n]+?)") , "$1::$2", 20),
40             // nameof(numbers)
41             // "numbers"
42             (new
43             → Regex(@"(?<before>\W)nameof\(((\r?\n)+\.)?(?<name>[a-zA-Z0-9_]+)(\r?\n)+)?\)",
44             → "${before}\"${name}\"", 0),
45             // Insert markers
46             // EqualityComparer<T> _equalityComparer = EqualityComparer<T>.Default;
47             // EqualityComparer<T> _equalityComparer =
48             → EqualityComparer<T>.Default; /*~_comparer~/
49             (new Regex(@"(?<declaration>EqualityComparer<(?<type>[\r?\n]+)>
50             → (?<comparer>[a-zA-Z0-9_]+) = EqualityComparer<k<type>>\.Default;)" ,
51             → "${declaration}/*~${comparer}~/", 0),
52             // /*~_equalityComparer~/..._equalityComparer.Equals(Minimum, value)
53             // /*~_equalityComparer~/...Minimum == value
54             (new Regex(@"(?<before>\/\s*(?<comparer>[a-zA-Z0-9_]+)~\s*(\r?\n)+\W)\k<comparer>\.Equ
55             → als\(((?<left>[\r?\n]+), (?<right>[\r?\n]+)\)", "${before}${left} == ${right}",
56             → 50),
57             // Remove markers
58             // /*~_equalityComparer~/
59             //
60             (new Regex(@"\r?\n[\r?\n]+\/\s*[a-zA-Z0-9_]+~\s*/") , "", 10),
61             // Insert markers
62             // Comparer<T> _comparer = Comparer<T>.Default;
63             // Comparer<T> _comparer = Comparer<T>.Default; /*~_comparer~/
64             (new Regex(@"(?<declaration>Comparer<(?<type>[\r?\n]+)> (?<comparer>[a-zA-Z0-9_]+) =
65             → Comparer<k<type>>\.Default;)" , "${declaration}/*~${comparer}~/", 0),
66             // /*~_comparer~/..._comparer.Compare(Minimum, value) <= 0
67             // /*~_comparer~/...Minimum <= value
68             (new Regex(@"(?<before>\/\s*(?<comparer>[a-zA-Z0-9_]+)~\s*(\r?\n)+\W)\k<comparer>\.Com
69             → pare\(((?<left>[\r?\n]+),
70             → (?<right>[\r?\n]+)\)\s*(?<comparison>[<=>]=?)\s*0(?<after>\D)" ,
71             → "${before}${left} ${comparison} ${right}${after}", 50),
72             // Remove markers
73             // private static readonly Comparer<T> _comparer =
74             → Comparer<T>.Default; /*~_comparer~/
75             //

```

```

63 (new Regex(@"r?\n[^\n]+\/*~[a-zA-Z0-9_]+~\*/", "", 10),
64 // Comparer<TArgument>.Default.Compare(maximumArgument, minimumArgument) < 0
65 // maximumArgument < minimumArgument
66 (new Regex(@"Comparer<[^>\n]+>\.Default\.Compare\\(s*(?<first>[^\n])\n+),\s*(?<second>
→ >[^\n]\n+)\s*)\\s*(?<comparison>[<>=]=?)\\s*0(?<after>\D)", "${first}
→ ${comparison} ${second} ${after}", 0),
67 // public static bool operator ==(Range<T> left, Range<T> right) =>
→ left.Equals(right);
68 //
69 (new Regex(@"r?\n[^\n]+bool operator ==\\((?<type>[^\n]+) (?<left>[a-zA-Z0-9_]+),
→ \k<type> (?<right>[a-zA-Z0-9_]+)\\) =>
→ ((\k<left>|\k<right>)\.Equals\\((\k<left>|\k<right>)\));", "", 10),
70 // public static bool operator !=(Range<T> left, Range<T> right) => !(left == right);
71 //
72 (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),
73 // public override bool Equals(object obj) => obj is Range<T> range ? Equals(range)
→ : false;
74 //
75 (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),
76 // out TProduct
77 // TProduct
78 (new Regex(@"(?<before><|, ))(in|out)
→ (?<typeParameter>[a-zA-Z0-9_]+)(?<after>>|,))",
→ "${before}${typeParameter}${after}", 10),
79 // public ...
80 // public: ...
81 (new Regex(@"(?<newLineAndIndent>r?\n?[
→ \t]*) (?<before>[^\{\\(\r\n)*] (?<access>private|protected|public) [ \t]+(?![^\{\\(\r\n
→ \n]*((?<=\\s)\\W)(interface|class|struct)(\\W)[^\{\\(\r\n)*[\\{\\(\r\n)])",
→ "${newLineAndIndent}${access}: ${before}", 0),
82 // public: static bool CollectExceptions { get; set; }
83 // public: inline static bool CollectExceptions;
84 (new Regex(@"(?<access>(private|protected|public): )(?<before>(static )?[^\r\n]+
→ )(?<name>[a-zA-Z0-9_]+) {[^;]}*(?<=\\W)get;[^;]}*(?<=\\W)set;[^;]}*")",
→ "${access}inline ${before}${name};", 0),
85 // public abstract class
86 // class
87 (new Regex(@"((public|protected|private|internal|abstract|static)
→ )*(?<category>interface|class|struct)", "${category}", 0),
88 // class GenericCollectionMethodsBase<TElement> {
89 // template <typename TElement> class GenericCollectionMethodsBase {
90 (new Regex(@"(?<before>r?\n)(?<indent>[ \t]*) (?<type>class|struct)
→ (?<typeName>[a-zA-Z0-9_]+)<(?<typeParameters>[a-zA-Z0-9
→ ,]+)>(?<typeDefinitionEnding>[^\{]+){", "${before}${indent}template <typename
→ ...> ${type} ${typeName};" + Environment.NewLine + "${indent}template <typename
→ ${typeParameters}> ${type}
→ ${typeName}<${typeParameters}>${typeDefinitionEnding}{", 0),
91 // static void
→ TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
→ tree, TElement* root)
92 // template<typename T> static void
→ TestMultipleCreationsAndDeletions<TElement>(SizedBinaryTreeMethodsBase<TElement>
→ tree, TElement* root)
93 (new Regex(@"static ([a-zA-Z0-9_]+) ([a-zA-Z0-9_]+)<([a-zA-Z0-9_]+)>\\(([^\\]\r\n)+)\\)",
→ "template <typename $3> static $1 $2($4)", 0),
94 // interface IFactory<out TProduct> {
95 // template <typename...> class IFactory; \ntemplate <typename TProduct> class
→ IFactory<TProduct>
96 (new Regex(@"(?<before>r?\n)(?<indent>[ \t]*)interface
→ (?<interface>[a-zA-Z0-9_]+)<(?<typeParameters>[a-zA-Z0-9
→ ,]+)>(?<typeDefinitionEnding>[^\{]+){", "${before}${indent}template <typename
→ ...> class ${interface};" + Environment.NewLine + "${indent}template <typename
→ ${typeParameters}> class
→ ${interface}<${typeParameters}>${typeDefinitionEnding}{", 0),
→ " public:", 0),
97 // template <typename TObject, TProperty, TValue>
98 // template <typename TObject, typename TProperty, typename TValue>
99 (new Regex(@"(?<before>template <((, )?typename [a-zA-Z0-9_]+)+,
→ )(?<typeParameter>[a-zA-Z0-9_]+)(?<after>(,|>))", "${before}typename
→ ${typeParameter}${after}", 10),
100 // Insert markers

```

```

101 // private: static void BuildExceptionString(this StringBuilder sb, Exception
102     ↳ exception, int level)
103 // /*~extensionMethod~BuildExceptionString~*/private: static void
104     ↳ BuildExceptionString(this StringBuilder sb, Exception exception, int level)
105 (new Regex(@"private: static [\r\n]+ (?<name>[a-zA-Z0-9]+)\(this [\r\n]+\)",
106     ↳ "/*~extensionMethod~${name}~*/$0", 0),
107 // Move all markers to the beginning of the file.
108 (new Regex(@"\A(?<before>[\r\n]+\r?\n(.|\n+)(?<marker>\/\*~extensionMethod~(?<name>
109     ↳ [a-zA-Z0-9]+)~\*/)", "${marker}${before}",
110     ↳ 10),
111 // /*~extensionMethod~BuildExceptionString~*/...sb.BuildExceptionString(exception.In
112     ↳ nerException, level +
113     ↳ 1);
114 // /*~extensionMethod~BuildExceptionString~*/...BuildExceptionString(sb,
115     ↳ exception.InnerException, level + 1);
116 (new Regex(@"(?<before>\/\*~extensionMethod~(?<name>[a-zA-Z0-9]+)~\*/(.|\n)+\W)(?<var
117     ↳ iable>[_a-zA-Z0-9]+\.\k<name>\(", "${before}${name}(${variable}",
118     ↳ 50),
119 // Remove markers
120 // /*~extensionMethod~BuildExceptionString~*/
121 //
122 (new Regex(@"\/\*~extensionMethod~[a-zA-Z0-9]+~\*/"), "", 0),
123 // (this
124 // (
125 (new Regex(@"\((this ", "(", 0),
126 // private: static readonly Disposal _emptyDelegate = (manual, wasDisposed) => { };
127 // private: inline static std::function<Disposal> _emptyDelegate = [](auto manual,
128     ↳ auto wasDisposed) { };
129 (new Regex(@"(?<access>(private|protected|public): )?static readonly
130     ↳ (?<type>[a-zA-Z][a-zA-Z0-9]*) (?<name>[a-zA-Z][a-zA-Z0-9_]*) =
131     ↳ \((?<firstArgument>[a-zA-Z][a-zA-Z0-9_]*)
132     ↳ (?<secondArgument>[a-zA-Z][a-zA-Z0-9_]*)\) => {\s*};"); "${access}inline static
133     ↳ std::function<${type}> ${name} = [](auto ${firstArgument}, auto
134     ↳ ${secondArgument}) { };", 0),
135 // public: static readonly EnsureAlwaysExtensionRoot Always = new
136     ↳ EnsureAlwaysExtensionRoot();
137 // public: inline static EnsureAlwaysExtensionRoot Always;
138 (new Regex(@"(?<access>(private|protected|public): )?static readonly
139     ↳ (?<type>[a-zA-Z0-9]+(<[a-zA-Z0-9]+>?)) (?<name>[a-zA-Z0-9_]*) = new
140     ↳ \k<type>\(\);", "${access}inline static ${type} ${name};", 0),
141 // public: static readonly Range<int> SByte = new
142     ↳ Range<int>(std::numeric_limits<int>::min(), std::numeric_limits<int>::max());
143 // public: inline static Range<int> SByte =
144     ↳ Range<int>(std::numeric_limits<int>::min(), std::numeric_limits<int>::max());
145 (new Regex(@"(?<access>(private|protected|public): )?static readonly
146     ↳ (?<type>[a-zA-Z0-9]+(<[a-zA-Z0-9]+>?)) (?<name>[a-zA-Z0-9_]*) = new
147     ↳ \k<type>\(\((?<arguments>[\r\n]+\)\);", "${access}inline static ${type} ${name} =
148     ↳ ${type}(${arguments});", 0),
149 // public: static readonly string ExceptionContentsSeparator = "---";
150 // public: inline static std::string ExceptionContentsSeparator = "---";
151 (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly) string
152     ↳ (?<name>[a-zA-Z0-9_]*) = ""(?:<string>\\\"|\"[\r\n]+)"";"), "${access}inline
153     ↳ static std::string ${name} = \"${string}\";", 0),
154 // private: const int MaxPath = 92;
155 // private: inline static const int MaxPath = 92;
156 (new Regex(@"(?<access>(private|protected|public): )?(const|static readonly)
157     ↳ (?<type>[a-zA-Z0-9]+) (?<name>[_a-zA-Z0-9]+) = (?<value>[~;\r\n]+);",
158     ↳ "${access}inline static const ${type} ${name} = ${value};", 0),
159 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
160     ↳ TArgument : class
161 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
162 (new Regex(@"(?<before> [a-zA-Z]+\((([a-zA-Z *],+ |)))(?<type>[a-zA-Z]+)(?<after>(
163     ↳ [a-zA-Z *],+)\))) [ \r\n]+where \k<type> : class", "${before}${type}*${after}",
164     ↳ 0),
165 // protected: abstract TElement GetFirst();
166 // protected: virtual TElement GetFirst() = 0;
167 (new Regex(@"(?<access>(private|protected|public): )?abstract
168     ↳ (?<method>[~;\r\n]+);", "${access}virtual ${method} = 0;", 0),
169 // TElement GetFirst();
170 // virtual TElement GetFirst() = 0;
171 (new Regex(@"(?<before>[\r\n]+ [ ]+)(?<methodDeclaration>(?!return) [a-zA-Z0-9]+
172     ↳ [a-zA-Z0-9]+\(([\r\n]*\))(?<after>; [ ]*\r\n+)", "${before}virtual
173     ↳ ${methodDeclaration} = 0${after}", 1),
174 // protected: readonly TreeElement[] _elements;
175 // protected: TreeElement _elements[N];

```

```

142 (new Regex(@"(?<access>(private|protected|public): )?readonly
    ↳ (?<type>[a-zA-Z<>0-9]+)([\\[]+) (?<name>[_a-zA-Z0-9]+);"), "${access}${type}
    ↳ ${name}[N];", 0),
143 // protected: readonly TElement Zero;
144 // protected: TElement Zero;
145 (new Regex(@"(?<access>(private|protected|public): )?readonly
    ↳ (?<type>[a-zA-Z<>0-9]+) (?<name>[_a-zA-Z0-9]+);"), "${access}${type} ${name};",
    ↳ 0),
146 // internal
147 //
148 (new Regex(@"(\\W)internal\\s+"), "$1", 0),
149 // static void NotImplementedException(ThrowExtensionRoot root) => throw new
    ↳ NotImplementedException();
150 // static void NotImplementedException(ThrowExtensionRoot root) { return throw new
    ↳ NotImplementedException(); }
151 (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),
152 // SizeBalancedTree(int capacity) => a = b;
153 // SizeBalancedTree(int capacity) { a = b; }
154 (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),
155 // int SizeBalancedTree(int capacity) => a;
156 // int SizeBalancedTree(int capacity) { return a; }
157 (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 $10; }", 0),
158 // OnDispose = (manual, wasDisposed) =>
159 // OnDispose = [&](auto manual, auto wasDisposed)
160 (new Regex(@"(?<variable>[a-zA-Z_][a-zA-Z0-9_]*)(?<operator>\\s*\\+?=\\s*)\\(((?<firstArg_
    ↳ ument>[a-zA-Z_][a-zA-Z0-9_]*),
    ↳ (?<secondArgument>[a-zA-Z_][a-zA-Z0-9_]*))\\s*=>"),
    ↳ "${variable}${operator}[&](auto ${firstArgument}, auto ${secondArgument})", 0),
161 // () => Integer<TElement>.Zero,
162 // () { return Integer<TElement>.Zero; },
163 (new Regex(@"\\(\\)\\s+=>\\s+(?<expression>[~() ;\\r\\n]+(\\(((?<parenthesis>\\()|(?<-parent_
    ↳ hesis>\\))| [~() ;\\r\\n]*?)))*? [~() ;\\r\\n]* (?<after>,|\\);)"), "()" { return
    ↳ ${expression}; }${after}", 0),
164 // ~DisposableBase() => Destruct();
165 // ~DisposableBase() { Destruct(); }
166 (new Regex(@"~(?<class>[a-zA-Z_][a-zA-Z0-9_]*)\\(\\)\\s+=>\\s+([~;\\r\\n]+?);"),
    ↳ "~${class}() { $1; }", 0),
167 // => Integer<TElement>.Zero;
168 // { return Integer<TElement>.Zero; }
169 (new Regex(@"\\)\\s+=>\\s+([~;\\r\\n]+?);"), ") { return $1; }", 0),
170 // () { return avlTree.Count; }
171 // [&]() -> auto { return avlTree.Count; }
172 (new Regex(@"(?<before>, |\\()\\(\\) { return (?<expression>[~;\\r\\n]+); }"),
    ↳ "${before}[&]() -> auto { return ${expression}; }", 0),
173 // Count => GetSizeOrZero(Root);
174 // Count() { return GetSizeOrZero(Root); }
175 (new Regex(@"(\\W)([A-Z][a-zA-Z]+)\\s+=>\\s+([~;\\r\\n]+);"), "$1$2() { return $3; }", 0),
176 // Insert scope borders.
177 // interface IDisposable { ... }
178 // interface IDisposable { /*~start~interface~IDisposable~*/ ...
    ↳ /*~end~interface~IDisposable~*/ }
179 (new Regex(@"(?<classDeclarationBegin>\\r?\\n(?<indent>[\\t ]*)interface[\\t
    ↳ ]*(?<type>[a-zA-Z_][a-zA-Z0-9_]*(\\<[^>\\n*>)? [~{]}*{) (?<middle>(\\.|\\n)*) (?<beforeE_
    ↳ nd>(\\<=\\r?\\n)\\k<indent>)(?<end>})"),
    ↳ "${classDeclarationBegin}/*~start~interface~${type}~*/${middle}${beforeEnd}/*~en_
    ↳ d~interface~${type}~*/${end}",
    ↳ 0),
180 // Inside the scope replace:
181 // /*~start~interface~IDisposable~*/ ... bool IsDisposed { get; } ...
    ↳ /*~end~interface~IDisposable~*/
182 // /*~start~interface~IDisposable~*/ ... virtual bool IsDisposed() = 0;
    ↳ /*~end~interface~IDisposable~*/
183 (new Regex(@"(?<before>(?<typeScopeStart>/\\*~start~interface~(?<type>[~^\\n\\*]+)~\\*/)
    ↳ (\\.|\\n)+)? (?<propertyDeclaration>(?<access>(private|protected|public):
    ↳ )?(?<propertyType>[a-zA-Z_][a-zA-Z0-9_]*) (?<property>[a-zA-Z_][a-zA-Z0-9_]*)
    ↳ (?<blockOpen>[\\n\\s]*{[\\n\\s]*)(\\[[^\\n\\+\\+][\\n\\s]*)?get; (?<blockClose>[\\n\\s]*)) (?<
    ↳ after>(\\.|\\n)+)? (?<typeScopeEnd>/\\*~end~interface~\\k<type>~\\*/))"),
    ↳ "${before}virtual ${propertyType} ${property}() = 0;${after}", 20),

```

```

184 // Remove scope borders.
185 // /*~start~interface~IDisposable~*/
186 //
187 (new Regex(@"\/\~([~\*\n]+)(~[~\*\n]+)*~\/"), "", 0),
188 // public: T Object { get; }
189 // public: const T Object;
190 (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),
191 // public: bool IsDisposed { get => _disposed > 0; }
192 // public: bool IsDisposed() { return _disposed > 0; }
193 (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),
194 // protected: virtual std::string ObjectName { get => GetType().Name; }
195 // protected: virtual std::string ObjectName() { return GetType().Name; }
196 (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),
197 // ArgumentInRange(string message) { string messageBuilder() { return message; }
198 // ArgumentInRange(string message) { auto messageBuilder = [&]() -> string { return
    → message; };
199 (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\n]*{(?<body>("[^"\\n]+|"[^"]*"|
    → [^"]*\n)+?)})")", "${before}auto ${methodName} = [&]() -> ${returnType}
    → {${body}};";", 10),
200 // Func<TElement> treeCount
201 // std::function<TElement()> treeCount
202 (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", 0),
203 // Action<TElement> free
204 // std::function<void(TElement)> free
205 (new Regex(@"Action<([?<typeParameters>[a-zA-Z0-9]+(,
    → ([a-zA-Z0-9]+))*)>)(?<after>>| (?<variable>[a-zA-Z0-9]+))"),
    → "std::function<void(${typeParameters})>${after}", 0),
206 // Predicate<TArgument> predicate
207 // std::function<bool(TArgument)> predicate
208 (new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<bool($1)>
    → $2", 0),
209 // var
210 // auto
211 (new Regex(@"(\\W)var(\\W)"), "$1auto$2", 0),
212 // unchecked
213 //
214 (new Regex(@"[\\r\\n]{2}\\s*unchecked\\s*${$}"), "", 0),
215 // throw new
216 // throw
217 (new Regex(@"(\\W)throw new(\\W)"), "$1throw$2", 0),
218 // void RaiseExceptionIgnoredEvent(Exception exception)
219 // void RaiseExceptionIgnoredEvent(const std::exception& exception)
220 (new Regex(@"(\\(| ) (System\\.Exception|Exception) (|\\))"), "$1const
    → std::exception&$3", 0),
221 // EventHandler<Exception>
222 // EventHandler<std::exception>
223 (new Regex(@"(\\W) (System\\.Exception|Exception) (\\W)"), "$1std::exception$3", 0),
224 // override void PrintNode(TElement node, StringBuilder sb, int level)
225 // void PrintNode(TElement node, StringBuilder sb, int level) override
226 (new Regex(@"override ([a-zA-Z0-9 \\*+]+)(\\([\\^\\r\\n]+?\\))"), "$1$2 override", 0),
227 // return (range.Minimum, range.Maximum)
228 // return {range.Minimum, range.Maximum}
229 (new Regex(@"(?<before>return\\s*)\\((?<values>[^\n]+)\n)\n)(?!\\n)(?<after>\\W)",
    → "${before}${values}${after}", 0),
230 // string
231 // std::string
232 (new Regex(@"(?<before>\\W)(?!::)string(?<after>\\W)"),
    → "${before}std::string${after}", 0),
233 // System.ValueTuple
234 // std::tuple

```

```

235 (new Regex(@"(?<before>\W) (System\.)?ValueTuple(?:\s*=\|() (?<after>\W)"),
    ↪ "${before}std::tuple${after}", 0),
236 // sbyte
237 // std::int8_t
238 (new Regex(@"(?<before>\W) ((System\.)?SB|sb)yte(?:\s*=\|() (?<after>\W)"),
    ↪ "${before}std::int8_t${after}", 0),
239 // short
240 // std::int16_t
241 (new Regex(@"(?<before>\W) ((System\.)?Int16|short) (?:\s*=\|() (?<after>\W)"),
    ↪ "${before}std::int16_t${after}", 0),
242 // int
243 // std::int32_t
244 (new Regex(@"(?<before>\W) ((System\.)?I|i)nt(32)?(?:\s*=\|() (?<after>\W)"),
    ↪ "${before}std::int32_t${after}", 0),
245 // long
246 // std::int64_t
247 (new Regex(@"(?<before>\W) ((System\.)?Int64|long) (?:\s*=\|() (?<after>\W)"),
    ↪ "${before}std::int64_t${after}", 0),
248 // byte
249 // std::uint8_t
250 (new Regex(@"(?<before>\W) ((System\.)?Byte|byte) (?:\s*=\|() (?<after>\W)"),
    ↪ "${before}std::uint8_t${after}", 0),
251 // ushort
252 // std::uint16_t
253 (new Regex(@"(?<before>\W) ((System\.)?UInt16|ushort) (?:\s*=\|() (?<after>\W)"),
    ↪ "${before}std::uint16_t${after}", 0),
254 // uint
255 // std::uint32_t
256 (new Regex(@"(?<before>\W) ((System\.)?UI|ui)nt(32)?(?:\s*=\|() (?<after>\W)"),
    ↪ "${before}std::uint32_t${after}", 0),
257 // ulong
258 // std::uint64_t
259 (new Regex(@"(?<before>\W) ((System\.)?UInt64|ulong) (?:\s*=\|() (?<after>\W)"),
    ↪ "${before}std::uint64_t${after}", 0),
260 // char*[] args
261 // char* args[]
262 (new Regex(@"([_a-zA-Z0-9:~*]?)\[\] ([a-zA-Z0-9]+)", "$1 $2[]", 0),
263 // float.MinValue
264 // std::numeric_limits<float>::lowest()
265 (new Regex(@"(?<before>\W) (?<type>std::[a-z0-9_]+|float|double)\.MinValue(?<after>\W)
    ↪ )"), "${before}std::numeric_limits<${type}>::lowest()${after}",
    ↪ 0),
266 // double.MaxValue
267 // std::numeric_limits<float>::max()
268 (new Regex(@"(?<before>\W) (?<type>std::[a-z0-9_]+|float|double)\.MaxValue(?<after>\W)
    ↪ )"), "${before}std::numeric_limits<${type}>::max()${after}",
    ↪ 0),
269 // using Platform.Numbers;
270 //
271 (new Regex(@"([\r\n]{2}|~)\s*using [\a-zA-Z0-9+;\s*?${"])", "", 0),
272 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
273 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
274 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(<[a-zA-Z0-9 ,]+>)? : ([a-zA-Z0-9]+)",
    ↪ "$1 $2$3 : public $4", 0),
275 // System.IDisposable
276 // System::IDisposable
277 (new Regex(@"(?<before>System(?:[a-zA-Z_]\w*)*)\. (?<after>[a-zA-Z_]\w*)",
    ↪ "${before}::${after}", 20),
278 // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
279 // class IProperty : public ISetter<TValue, TObject>, public IProvider<TValue,
    ↪ TObject>
280 (new Regex(@"(?<before>(interface|struct|class) [a-zA-Z_]\w* : ((public
    ↪ [a-zA-Z_]\w*:(<[a-zA-Z0-9 ,]+>)?,
    ↪ )+)?(?<inheritedType>(?!public)[a-zA-Z_]\w*:(<[a-zA-Z0-9 ,]+>)?(?<after>(,
    ↪ [a-zA-Z_]\w*:(!>)|[\r\n]+))", "${before}public ${inheritedType}${after}",
    ↪ 10),
281 // interface IDisposable {
282 // class IDisposable { public:
283 (new Regex(@"(?<before>\r?\n) (?<indent>[ \t]*)interface
    ↪ (?<interface>[a-zA-Z_]\w*) (?<typeDefinitionEnding>[~{}+]{")
    ↪ "${before}${indent}class ${interface}${typeDefinitionEnding}{ " +
    ↪ Environment.NewLine + " public:", 0),
284 // struct TreeElement { }
285 // struct TreeElement { };
286 (new Regex(@"(struct|class) ([a-zA-Z0-9]+) (\s+){([\sa-zA-Z0-9;:_]+?)}([~;])", "$1
    ↪ $2$3{$4};$5", 0),

```

```

287 // class Program { }
288 // class Program { };
289 (new Regex(@"(?<type>struct|class)
→ (?<name>[a-zA-Z0-9]+[\r\n]*) (?<beforeBody>[\r\n]+(?<indentLevel>[\t
→ ]*))? \{ (?<body>[\\s]+?[\r\n]+\k<indentLevel>\} (?<afterBody>[;]|$)"), "${type}
→ ${name}${beforeBody}${body}${afterBody}", 0),
290 // Insert scope borders.
291 // ref TElement root
292 // ~!root!~ref TElement root
293 (new Regex(@"(?<definition>(?!<pointer>[a-zA-Z0-9]+| [a-zA-Z0-9]+(?<ref>ref))
→ (?<variable>[a-zA-Z0-9]+)(?<value>=|,| =)")), "~!${variable}!~!${definition}", 0),
294 // Inside the scope of ~!root!~ replace:
295 // root
296 // *root
297 (new Regex(@"(?<definition>~!(?<pointer>[a-zA-Z0-9]+)!~ref [a-zA-Z0-9]+
→ \k<pointer>(?!<value>=|,| =) (?<before>((?!~!\k<pointer>!) (.\n))*?) (?<prefix>(\W
→ | \() \k<pointer> (?<suffix> ( | \) | ; | , ))")),
→ "${definition}${before}${prefix}*${pointer}${suffix}", 70),
298 // Remove scope borders.
299 // ~!root!~
300 //
301 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", 5),
302 // ref auto root = ref
303 // ref auto root =
304 (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)"), "$1* $2 =$3", 0),
305 // *root = ref left;
306 // root = left;
307 (new Regex(@"\*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)"), "$1 = $2$3", 0),
308 // (ref left)
309 // (left)
310 (new Regex(@"\ (ref ([a-zA-Z0-9]+) (\) | \ (|,))"), "($1$2", 0),
311 // ref TElement
312 // TElement*
313 (new Regex(@"( | \() ref ([a-zA-Z0-9]+) "), "$1$2* ", 0),
314 // ref sizeBalancedTree.Root
315 // &sizeBalancedTree->Root
316 (new Regex(@"ref ([a-zA-Z0-9]+) \. ([a-zA-Z0-9\*]+)"), "&$1->$2", 0),
317 // ref GetElement(node).Right
318 // &GetElement(node)->Right
319 (new Regex(@"ref ([a-zA-Z0-9]+) \ ( ([a-zA-Z0-9\*]+) \) \. ([a-zA-Z0-9]+)"),
→ "&$1($2)->$3", 0),
320 // GetElement(node).Right
321 // GetElement(node)->Right
322 (new Regex(@"([a-zA-Z0-9]+) \ ( ([a-zA-Z0-9\*]+) \) \. ([a-zA-Z0-9]+)"), "$1($2)->$3", 0),
323 // [Fact] \npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
324 // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
325 (new Regex(@"\[Fact\] \[s\n\]+(public:)?(static)?void ([a-zA-Z0-9]+) \ ( \)"), "public:
→ TEST_METHOD($3)", 0),
326 // class TreesTests
327 // TEST_CLASS(TreesTests)
328 (new Regex(@"class ([a-zA-Z0-9]+Tests)"), "TEST_CLASS($1)", 0),
329 // Assert.Equal
330 // Assert::AreEqual
331 (new Regex(@"(?<type>Assert) \. (?<method>(Not)?Equal)"), "${type}::Are${method}", 0),
332 // Assert.Throws
333 // Assert::ExpectException
334 (new Regex(@"(Assert) \. Throws"), "$1::ExpectException", 0),
335 // Assert.True
336 // Assert::IsTrue
337 (new Regex(@"(Assert) \. (True|False)"), "$1::Is$2", 0),
338 // $"Argument {argumentName} is null."
339 // std::string("Argument
→ ").append(Platform::Converters::To<std::string>(argumentName)).append(" is
→ null.")
340 (new Regex(@"\$"" (?<left>(\\" | [^""\r\n])*) { (?<expression>[_a-zA-Z0-9]+) } { (?<right>(\\"
→ | "" | [^""\r\n])*) """,
→ "std::string(\$\"${left}\").append(Platform::Converters::To<std::string>(${expres
→ sion})).append(\"${right}\")",
→ 10),
341 // $"
342 // "
343 (new Regex(@"\$"""), "\"", 0),
344 // std::string(std::string("[").append(Platform::Converters::To<std::string>(Minimum
→ )), append("
→ ").append(Platform::Converters::To<std::string>(Maximum)).append("]")
345 // std::string("[").append(Platform::Converters::To<std::string>(Minimum)).append("
→ ").append(Platform::Converters::To<std::string>(Maximum)).append("]")

```



```

346 (new Regex(@"std::string\((?<begin>std::string\(\"\"\\\"|\"\"[^\"]*)\"\"\\)\.append\((Platf
    ↪ orm::Converters::To<std::string>\([^\n]+\)|[^\n]+\))\)\.append\"),
    ↪ \"${begin}.append\", 10),
347 // Console.WriteLine(\"...\")
348 // printf(\"...\n\")
349 (new Regex(@"Console.WriteLine\(\"\"([^\r\n]+)\"\"\\)\", \"printf(\"$1\\n\\n\")\", 0),
350 // TElement Root;
351 // TElement Root = 0;
352 (new Regex(@"(?<before>\r?\n[\t ]+)(?<access>(private|protected|public)(:
    ↪ )?)?(?<type>[a-zA-Z0-9:~_]+(?<!return>)) (?<name>[_a-zA-Z0-9~_]+);\"),
    ↪ \"${before}${access}${type} ${name} = 0;\", 0),
353 // TreeElement _elements[N];
354 // TreeElement _elements[N] = { {0} };
355 (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),
356 // auto path = new TElement[MaxPath];
357 // TElement path[MaxPath] = { {0} };
358 (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),
359 // bool Equals(Range<T> other) { ... }
360 // bool operator ==(const Key &other) const { ... }
361 (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),
362 // Insert scope borders.
363 // class Range { ... public: override std::string ToString() { return ...; }
364 // class Range { /*~Range<T>~*/ ... public: override std::string ToString() { return
    ↪ ...; }
365 (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)template <typename
    ↪ (?<typeParameter>[^\n~]+> (struct|class)
    ↪ (?<type>[_a-zA-Z0-9~_]+<~<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),
366 // Inside the scope of ~!Range!~ replace:
367 // public: override std::string ToString() { return ...; }
368 // public: operator std::string() const { return ...; }~\n~public: friend
    ↪ std::ostream & operator <<(std::ostream &out, const A &obj) { return out <<
    ↪ (std::string)obj; }
369 (new Regex(@"(?<scope>/\s*(?<type>[_a-zA-Z0-9~_]+>~\s*/)(?<separator>.\n)(?<before>
    ↪ ((?!/\s*~\s*<type>~\s*/)(.\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
    ↪ ${type} &obj) { return out << (std::string)obj; }\", 0),
370 // Remove scope borders.
371 // /*~Range~*/
372 //
373 (new Regex(@"/\s*~[_a-zA-Z0-9~_]+>~\s*/\"), \"\", 0),
374 // private: inline static ConcurrentBag<std::exception> _exceptionsBag;
375 // private: inline static std::mutex _exceptionsBag_mutex; \n\n private: inline
    ↪ static std::vector<std::exception> _exceptionsBag;
376 (new Regex(@"(?<begin>\r?\n(?<indent>[\t ]+))?(?<access>(private|protected|public):
    ↪ )?inline static ConcurrentBag<(?(<argumentType>[^\r\n~]+)>
    ↪ (?<name>[_a-zA-Z0-9~_]+);\"), \"${begin}private: inline static std::mutex
    ↪ ${name}_mutex;\" + Environment.NewLine + Environment.NewLine +
    ↪ \"${indent}${access}inline static std::vector<${argumentType}> ${name};\", 0),
377 // public: static IReadOnlyCollection<std::exception> GetCollectedExceptions() {
    ↪ return _exceptionsBag; }
378 // public: static std::vector<std::exception> GetCollectedExceptions() { return
    ↪ std::vector<std::exception>(_exceptionsBag); }
379 (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),
380 // public: static event EventHandler<std::exception> ExceptionIgnored =
    ↪ OnExceptionIgnored; ... };
381 // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
    ↪ const std::exception&> ExceptionIgnored = OnExceptionIgnored; };

```



```

382 (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),
383 // public: event Disposal OnDispose;
384 // public: Platform::Delegates::MulticastDelegate<Disposal> OnDispose;
385 (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}> ${name};", 0),
386 // Insert scope borders.
387 // class IgnoredExceptions { ... private: inline static std::vector<std::exception>
    ↳ _exceptionsBag;
388 // class IgnoredExceptions {/~_exceptionsBag~/ ... private: inline static
    ↳ std::vector<std::exception> _exceptionsBag;
389 (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n}]+\r\n[\t
    ↳ ]*(?<middle>((?!class).\n)+?) (?<vectorFieldDeclaration>(?<access>(private|pro
    ↳ tected|public): )inline static std::vector<(?<argumentType>[~;\r\n]+)>
    ↳ (?<fieldName>[_a-zA-Z0-9]+);)"),
    ↳ "${classDeclarationBegin}/~${fieldName}~/${middle}${vectorFieldDeclaration}",
    ↳ 0),
390 // Inside the scope of ~!_exceptionsBag!~ replace:
391 // _exceptionsBag.Add(exception);
392 // _exceptionsBag.push_back(exception);
393 (new Regex(@"(?<scope>/~*(?<fieldName>[_a-zA-Z0-9]+)~*/) (?<separator>.\n) (?<befor
    ↳ e>((?!/~*\k<fieldName>~*/)(.\n))*?)\k<fieldName>\.Add"),
    ↳ "${scope}${separator}${before}${fieldName}.push_back", 10),
394 // Remove scope borders.
395 // /*~_exceptionsBag~/
396 //
397 (new Regex(@"/*~[_a-zA-Z0-9]+~*/"), "", 0),
398 // Insert scope borders.
399 // class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
400 // class IgnoredExceptions {/~_exceptionsBag~/ ... private: static std::mutex
    ↳ _exceptionsBag_mutex;
401 (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)class [^{\r\n}]+\r\n[\t
    ↳ ]*(?<middle>((?!class).\n)+?) (?<mutexDeclaration>private: inline static
    ↳ std::mutex (?<fieldName>[_a-zA-Z0-9]+) _mutex;)",
    ↳ "${classDeclarationBegin}/~${fieldName}~/${middle}${mutexDeclaration}", 0),
402 // Inside the scope of ~!_exceptionsBag!~ replace:
403 // return std::vector<std::exception>(_exceptionsBag);
404 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
    ↳ std::vector<std::exception>(_exceptionsBag);
405 (new Regex(@"(?<scope>/~*(?<fieldName>[_a-zA-Z0-9]+)~*/) (?<separator>.\n) (?<befor
    ↳ e>((?!/~*\k<fieldName>~*/)(.\n))*?) { (?<after>((?!lock_guard)[^{};\r\n])*\k<f
    ↳ ieldName>[~;}\r\n]*);)", "${scope}${separator}${before}{
    ↳ std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", 10),
406 // Inside the scope of ~!_exceptionsBag!~ replace:
407 // _exceptionsBag.Add(exception);
408 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
    ↳ _exceptionsBag.Add(exception);
409 (new Regex(@"(?<scope>/~*(?<fieldName>[_a-zA-Z0-9]+)~*/) (?<separator>.\n) (?<befor
    ↳ e>((?!/~*\k<fieldName>~*/)(.\n))*?) { (?<after>((?!lock_guard)([~{};]|\n))*?\r
    ↳ ?\n(?<indent>[\t ]*)\k<fieldName>[~;}\r\n]*);)",
    ↳ "${scope}${separator}${before}{
    ↳ "${indent}std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", 10),
410 // Remove scope borders.
411 // /*~_exceptionsBag~/
412 //
413 (new Regex(@"/*~[_a-zA-Z0-9]+~*/"), "", 0),
414 // Insert scope borders.
415 // class IgnoredExceptions { ... public: static inline
    ↳ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
    ↳ ExceptionIgnored = OnExceptionIgnored;
416 // class IgnoredExceptions {/~ExceptionIgnored~/ ... public: static inline
    ↳ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
    ↳ ExceptionIgnored = OnExceptionIgnored;
417 (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),

```

```

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

```

```

(new Regex(@"if \((?<variable>[a-zA-Z0-9]+)\.insert\((?<argument>[a-zA-Z0-9]+)\)\)(?
  <separator>[\t ]*[\r\n]+)(?<indent>[\t ]*){", "if
  → (!${variable}.contains(${argument}))${separator}${indent}{ " +
  → Environment.NewLine + "${indent}    ${variable}.insert(${argument});", 0),
  // Remove scope borders.
  // ~!added!~
  //
  (new Regex(@"~![a-zA-Z0-9]+!~"), "", 5),
  // Insert scope borders.
  // auto random = new System::Random(0);
  // std::srand(0);
  (new Regex(@"[a-zA-Z0-9\.] + ([a-zA-Z0-9]+) = new
  → (System::)?Random\(((?<argument>[a-zA-Z0-9]+)\);", "~!$1!~std::srand($3);", 0),
  // Inside the scope of ~!random!~ replace:
  // random.Next(1, N)
  // (std::rand() % N) + 1
  (new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>[.\n])(?<before>((?<
  → !~!\k<variable>!~)([.\n])*)\k<variable>\.Next\(((?<from>[a-zA-Z0-9]+),
  → (?<to>[a-zA-Z0-9]+)\)", "${scope}${separator}${before}(std::rand() % ${to}) +
  → ${from}", 10),
  // Remove scope borders.
  // ~!random!~
  //
  (new Regex(@"~![a-zA-Z0-9]+!~"), "", 5),
  // Insert method body scope starts.
  // void PrintNodes(TElement node, StringBuilder sb, int level) {
  // void PrintNodes(TElement node, StringBuilder sb, int level) { /*method-start*/
  (new Regex(@"(?<start>\r?\n[\t ]*)(?<prefix>((private|protected|public): )?(virtual
  → )?[a-zA-Z0-9:_]+
  → )?(?<method>[a-zA-Z][a-zA-Z0-9_]*)((?<arguments>[^\)]*)\)(?<override>(
  → override)?)(?<separator>[\t\r\n]*)\{(?<end>[~])\"", "${start}${prefix}${method}
  → (${arguments})${override}${separator}{ /*method-start*/${end} ",
  → 0),
  // Insert method body scope ends.
  // { /*method-start*/... }
  // { /*method-start*/... /*method-end*/ }
  (new Regex(@"{\/*method-start\*/(?<body>((?<bracket>\{) | (?<-bracket>\}) | [^\{\}])*)+ )
  → \}"}", "{ /*method-start*/${body} /*method-end*/ }",
  → 0),
  // Inside method bodies replace:
  // GetFirst(
  // this->GetFirst(
  (new
  → Regex(@"(?<scope>\/*method-start\*/)(?<before>((?!\/\*method-end\*/)([.\n])*)?(?
  → <separator>[\W](?!(:|\.|->|throw\s+)))(?<method>(?!sizeof)[a-zA-Z0-9_]*)((?!\
  → \{)(?<after>([.\n])*)?(?<scopeEnd>\/*method-end\*/)",
  → "${scope}${before}${separator}this->${method}(${after}${scopeEnd}", 100),
  // Remove scope borders.
  // /*method-start*/
  //
  (new Regex(@"\/\*method-(start|end)\*/"), "", 0),
  // Insert scope borders.
  // const std::exception& ex
  // const std::exception& ex/*~ex~*/
  (new Regex(@"(?<before>\(| |)(?<variableDefinition>(const )?(std::)?exception&?
  → (?<variable>[_a-zA-Z0-9_])?(?<after>\W)",
  → "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
  // Inside the scope of ~!ex!~ replace:
  // ex.Message
  // ex.what()
  (new Regex(@"(?<scope>\/*~(?<variable>[_a-zA-Z0-9_]+)~\*/)(?<separator>[.\n])(?<before
  → >((?!\/\*~\k<variable>~\*/)([.\n])*)?(Platform::Converters::To<std::string>\(\k<
  → variable>\.Message\)|\k<variable>\.Message)",
  → "${scope}${separator}${before}${variable}.what()", 10),
  // Remove scope borders.
  // /*~ex~*/
  //
  (new Regex(@"\/\*~[_a-zA-Z0-9_]+~\*/"), "", 0),
  // throw ObjectDisposedException(objectName, message);
  // throw std::runtime_error(std::string("Attempt to access disposed object
  → ").append(objectName).append(": ").append(message).append("."));
  (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),

```

```

512 // throw ArgumentNullException(argumentName, message);
513 // throw std::invalid_argument(std::string("Argument
514 → ").append(argumentName).append(" is null: ").append(message).append("."));
515 (new Regex(@"throw
516 → ArgumentNullException\((?<argument>[a-zA-Z]*[Aa]rgument[a-zA-Z]*),
517 → (?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\))?\)", "throw
518 → std::invalid_argument(std::string(\"Argument \").append(${argument}).append(\"
519 → is null: \").append(${message}).append(\".\"));", 0),
520 // throw ArgumentException(message, argumentName);
521 // throw std::invalid_argument(std::string("Invalid ").append(argumentName).append("
522 → argument: ").append(message).append("."));
523 (new Regex(@"throw
524 → ArgumentException\((?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\))?),
525 → (?<argument>[a-zA-Z]*[Aa]rgument[a-zA-Z]*\)", "throw
526 → std::invalid_argument(std::string(\"Invalid \").append(${argument}).append(\"
527 → argument: \").append(${message}).append(\".\"));", 0),
528 // throw ArgumentOutOfRangeException(argumentName, argumentValue, messageBuilder());
529 // throw std::invalid_argument(std::string("Value
530 → [").append(Platform::Converters::To<std::string>(argumentValue)).append("] of
531 → argument [").append(argumentName).append("] is out of range:
532 → ").append(messageBuilder()).append("."));
533 (new Regex(@"throw ArgumentOutOfRangeException\((?<argument>[a-zA-Z]*[Aa]rgument[a-z
534 → A-Z]*([Nn]ame[a-zA-Z]*)?),
535 → (?<argumentValue>[a-zA-Z]*[Aa]rgument[a-zA-Z]*([Vv]alue[a-zA-Z]*)?),
536 → (?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*(\(\))?\)", "throw
537 → std::invalid_argument(std::string(\"Value
538 → [").append(Platform::Converters::To<std::string>(${argumentValue})).append(\"
539 → of argument [").append(${argument}).append(\"] is out of range:
540 → \").append(${message}).append(\".\"));", 0),
541 // throw NotSupportedException();
542 // throw std::logic_error("Not supported exception.");
543 (new Regex(@"throw NotSupportedException\(\);", "throw std::logic_error(\"Not
544 → supported exception.\");", 0),
545 // throw NotImplementedException();
546 // throw std::logic_error("Not implemented exception.");
547 (new Regex(@"throw NotImplementedException\(\);", "throw std::logic_error(\"Not
548 → implemented exception.\");", 0),
549 // Insert scope borders.
550 // const std::string& message
551 // const std::string& message/*~message~*/
552 (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?string&?|char\*)
553 → (?<variable>[_a-zA-Z0-9]+)(?<after>\W)",
554 → "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
555 // Inside the scope of /*~message~*/ replace:
556 // Platform::Converters::To<std::string>(message)
557 // message
558 (new Regex(@"(?<scope>\/\*~(?<variable>[_a-zA-Z0-9]+)~\*\/)(?<separator>.\|\\n)(?<before>
559 → >((?!\/\*~\k<variable>~\*\/)(.\|\\n))*?)Platform::Converters::To<std::string>\(\k<v
560 → ariable>\)", "${scope}${separator}${before}${variable}",
561 → 10),
562 // Remove scope borders.
563 // /*~ex~*/
564 //
565 (new Regex(@"\/\*~[_a-zA-Z0-9]+~\*\/", "", 0),
566 // Insert scope borders.
567 // std::tuple<T, T> tuple
568 // std::tuple<T, T> tuple/*~tuple~*/
569 (new Regex(@"(?<before>\(| )(?<variableDefinition>(const )?(std::)?tuple<[^\\n]+>&?
570 → (?<variable>[_a-zA-Z0-9]+)(?<after>\W)",
571 → "${before}${variableDefinition}/*~${variable}~*/${after}", 0),
572 // Inside the scope of ~!ex!~ replace:
573 // tuple.Item1
574 // std::get<1-1>(tuple)
575 (new Regex(@"(?<scope>\/\*~(?<variable>[_a-zA-Z0-9]+)~\*\/)(?<separator>.\|\\n)(?<before>
576 → >((?!\/\*~\k<variable>~\*\/)(.\|\\n))*?)\k<variable>\.Item(?<itemNumber>\d+)(?<afte
577 → r>\W)",
578 → "${scope}${separator}${before}std::get<${itemNumber}-1>(${variable})${after}",
579 → 10),
580 // Remove scope borders.
581 // /*~ex~*/
582 //
583 (new Regex(@"\/\*~[_a-zA-Z0-9]+~\*\/", "", 0),
584 // Insert scope borders.
585 // class Range<T> {
586 // class Range<T> { /*~type~Range<T>~*/

```

```

554 (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),
555 // Inside the scope of /*~type~Range<T>~*/ insert inner scope and replace:
556 // public: static implicit operator std::tuple<T, T>(Range<T> range)
557 // public: operator std::tuple<T, T>() const { /*~variable~Range<T>~*/
558 (new Regex(@"(?<scope>/\s*~type~(?(<typeName>[^\n\*]+)~(?(<fullType>[^\n\*]+)~\s*/)(?<
    ↳ separator>.\n)(?<before>((?!/\s*~type~\k<typeName>~\k<fullType>~\s*/)(.\n))*?)(
    ↳ ?<access>(private|protected|public): )static implicit operator
    ↳ (?<targetType>[^\n\*]+)\((?<argumentDeclaration>\k<fullType>
    ↳ (?<variable>[a-zA-Z0-9]+))\)(?<after>\s*\n?~{n})"),
    ↳ "$${scope}${separator}${before}${access}operator ${targetType}()
    ↳ const${after}/*~variable~${variable}~*/", 10),
559 // Inside the scope of /*~type~Range<T>~*/ replace:
560 // 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)); }
561 // public: Range(std::tuple<T, T> tuple) : Range(std::get<1-1>(tuple),
    ↳ std::get<2-1>(tuple)) { }
562 (new Regex(@"(?<scope>/\s*~type~(?(<typeName>[^\n\*]+)~(?(<fullType>[^\n\*]+)~\s*/)(?<
    ↳ separator>.\n)(?<before>((?!/\s*~type~\k<typeName>~\k<fullType>~\s*/)(.\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),
563 // Inside the scope of /*~variable~range~*/ replace:
564 // range.Minimum
565 // this->Minimum
566 (new Regex(@"(?<scope>{\s*~variable~(?(<variable>[^\n\*]+)~\s*/)(?<separator>.\n)(?<be
    ↳ fore>(?(<beforeExpression>(?(<bracket>{) | (?(<-bracket>}) | [^{}]| \n)*?)\k<variable>\.
    ↳ (?<field>[_a-zA-Z0-9]+)(?<after>(, | ; | |
    ↳ | \)))(?<afterExpression>(?(<bracket>{) | (?(<-bracket>}) | [^{}]| \n)*?)"),
    ↳ "$${scope}${separator}${before}this->${field}${after}", 10),
567 // Remove scope borders.
568 // /*~ex~*/
569 //
570 (new Regex(@"/*~[^\n\*]+~[^\n\*]+~*/"), "", 0),
571 // Insert scope borders.
572 // namespace Platform::Ranges { ... }
573 // namespace Platform::Ranges { /*~start~namespace~Platform::Ranges~*/ ...
    ↳ /*~end~namespace~Platform::Ranges~*/ }
574 (new Regex(@"(?<namespaceDeclarationBegin>\r?\n(?<indent>[\t ]*)namespace
    ↳ (?<namespaceName>(?(<namePart>[a-zA-Z][a-zA-Z0-9]+)(?<nextNamePart>::[a-zA-Z][a-z
    ↳ A-Z0-9]+))(\s|\n)*}{(?(<middle>(\s|\n)*)(?<end>(?(<=\\r?\n)\k<indent>}{?!;))}"),
    ↳ "$${namespaceDeclarationBegin}/*~start~namespace~${namespaceName}~*/${middle}/*~e
    ↳ nd~namespace~${namespaceName}~*/${end}",
    ↳ 0),
575 // Insert scope borders.
576 // class Range<T> { ... };
577 // class Range<T> { /*~start~type~Range<T>~T~*/ ... /*~end~type~Range<T>~T~*/ };
578 (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>(\s|\n)*)(?<endIndent>(?(<=\\r?\n)\k<indent>){?<end>};)"),
    ↳ "$${classDeclarationBegin}/*~start~type~${type}~${typeParameter}~*/${middle}${end}
    ↳ Indent}/*~end~type~${type}~${typeParameter}~*/${end}",
    ↳ 0),
579 // Inside the scope replace:
580 // /*~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~*/
581 // /*~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(); } }; }

```

```

(new Regex(@"(?<namespaceScopeStart>/\~*start~namespace~(?<namespace>[~\n\*]+)~\*/)
  (?<betweenStartScopes>(.|\n)+)(?<typeScopeStart>/\~*start~type~(?<type>[~\n\*]+)
  )~(?<typeParameter>[~\n\*]+)~\*/)(?<before>(.|\n)+)?(?<hashMethodDeclaration>\r
  ↪ ?\n[ \t]*(?<access>(private|protected|public): )override std::int32_t
  ↪ GetHashCode\(\) (\s|\n)*{\s*(?<methodBody>[~\s][~\n]+[~\s])\s*\s*(?<after>(.|\n
  ↪ )+)?(?<typeScopeEnd>/\~*end~type~\k<type>~\k<typeParameter>~\*/)(?<betweenEndSco
  ↪ pes>(.|\n)+)(?<namespaceScopeEnd>/\~*end~namespace~\k<namespace>~\*/)}r?\n"),
  ↪ "${namespaceScopeStart}${betweenStartScopes}${typeScopeStart}${before}${after}${
  ↪ typeScopeEnd}${betweenEndScopes}${namespaceScopeEnd}" + Environment.NewLine +
  ↪ Environment.NewLine + "namespace std" + Environment.NewLine + "{" +
  ↪ Environment.NewLine + "    template <typename ${typeParameter}>" +
  ↪ Environment.NewLine + "    struct hash<${namespace}:${type}>" +
  ↪ Environment.NewLine + "    {" + Environment.NewLine + "        std::size_t
  ↪ operator()(const ${namespace}:${type} &obj) const" + Environment.NewLine + "
  ↪    {" + Environment.NewLine + "
  ↪    /*~start~method~*/${methodBody}/*~end~method~*/" + Environment.NewLine + "
  ↪    }" + Environment.NewLine + "    };" + Environment.NewLine + "}" +
  ↪ Environment.NewLine, 10),
// Inside scope of /*~start~method~*/ replace:
// /*~start~method~*/ ... Minimum ... /*~end~method~*/
// /*~start~method~*/ ... obj.Minimum ... /*~end~method~*/
583 (new Regex(@"(?<methodScopeStart>/\~*start~method~\*/)(?<before>.+({|,
  ↪ ))(?<name>[a-zA-Z][a-zA-Z0-9]+)(?<after>[~\n\.\(a-zA-Z0-9)((?!/\~*end~method~\*/|
  ↪ ) [~\n])+)(?<methodScopeEnd>/\~*end~method~\*/)"),
  ↪ "${methodScopeStart}${before}obj.${name}${after}${methodScopeEnd}", 10),
587 // Remove scope borders.
588 // /*~start~type~Range<T>~*/
589 //
590 (new Regex(@"/*~[~\*~\n]+(~[~\*~\n]+)*~\*/"), "", 0),
591 // class Disposable<T> : public Disposable
592 // class Disposable<T> : public Disposable<>
593 (new Regex(@"(?<before>(struct|class) (?<type>[a-zA-Z][a-zA-Z0-9]*)<[~<>\n]+> :
  ↪ (?<access>(private|protected|public) )?\k<type>)(?<after>\b(?:<))"),
  ↪ "${before}<>${after}", 0),
594 // Insert scope borders.
595 // class Disposable<T> : public Disposable<> { ... };
596 // class Disposable<T> : public Disposable<>
  ↪ { /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/ ...
  ↪ /*~end~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/ };
597 (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)template[\t
  ↪ ]*(?<typeParameters>[~\n\*])>[\t ]*(struct|class)[\t
  ↪ ]+(?<fullType>(?(type>[a-zA-Z][a-zA-Z0-9]*)<[~<>\n]*>)?)[\t ]*:[\t
  ↪ ]*(?<access>(private|protected|public) [\t
  ↪ ]+)?(?(fullBaseType>(?(baseType>[a-zA-Z][a-zA-Z0-9]*)<[~<>\n]*>)?)[\t
  ↪ ]*(\r?\n)?[\t
  ↪ ]*{ } (?(middle>(.|\n)*)(?(beforeEnd>(?(=\r?\n)\k<indent>)(?<end>};))"),
  ↪ "${classDeclarationBegin}/*~start~type~${type}~${fullType}~${baseType}~${fullBas
  ↪ eType}~*/${middle}${beforeEnd}/*~end~type~${type}~${fullType}~${baseType}~${full
  ↪ BaseType}~*/${end}",
  ↪ 0),
598 // Inside the scope replace:
599 // /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/ ... ) : base(
  ↪ ... /*~end~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/
600 // /*~start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/ ... ) :
  ↪ Disposable<>( /*~end~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/
601 (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),
602 // Inside the scope replace:
603 // /*~start~type~Disposable~Disposable<T>~X~X<>~*/ ... ) : base( ...
  ↪ /*~end~type~Disposable~Disposable<T>~X~X<>~*/
604 // /*~start~type~Disposable~Disposable<T>~X~X<>~*/ ... ) : X(
  ↪ /*~end~type~Disposable~Disposable<T>~X~X<>~*/
605 (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),
606 // Inside the scope replace:
607 // /*~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<>~*/
608 // /*~start~type~Disposable~Disposable<T>~X~X<>~*/ ... public: Disposable(T object)
  ↪ { Object = object; } /*~end~type~Disposable~Disposable<T>~X~X<>~*/

```

```

609 (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)+?(?<duplicateConstru
    ↪ ctor>(?(access>(private|protected|public):[\t
    ↪ ]*)?k<type>\(k<arguments>\)\s*: [~{}\n]+\s*{[~{}\n]+}) (?(after>(.|\n)+?(?<typeS
    ↪ copeEnd>/\~*end~type~k<types>~\*/))"), "{$before}$after}",
    ↪ 20),
610 // Remove scope borders.
611 // /\~*start~type~Disposable~Disposable<T>~Disposable~Disposable<>~*/
612 //
613 (new Regex(@"/\~*[~\*\n]+(~[~\*\n]+)*~\*/"), "", 0),
614 // Insert scope borders.
615 // private: inline static const AppDomain _currentDomain = AppDomain.CurrentDomain;
616 // private: inline static const AppDomain _currentDomain =
    ↪ AppDomain.CurrentDomain; /\~*app-domain~_currentDomain~*/
617 (new Regex(@"(?<declaration>(?(access>(private|protected|public):[\t ]*)?(inline[\t
    ↪ ]+)?(static[\t ]+)?(const[\t ]+)?AppDomain[\t
    ↪ ]+(?<field>[a-zA-Z_][a-zA-Z0-9_]*)[\t ]*=[\t ]*AppDomain\.CurrentDomain;")),
    ↪ "{$declaration}/\~*app-domain~_{field}~*/", 0),
618 // Inside the scope replace:
619 // /\~*app-domain~_currentDomain~*/ ... _currentDomain.ProcessExit += OnProcessExit;
620 // /\~*app-domain~_currentDomain~*/ ... std::atexit(OnProcessExit);
621 (new Regex(@"(?<before>(?<fieldScopeStart>/\~*app-domain~(?<field>[~\n\*]+)~\*/)(.|\n
    ↪ )+?)k<field>\.ProcessExit[\t ]*\+=[\t
    ↪ ]*(?<eventHandler>[a-zA-Z_][a-zA-Z0-9_]*)"); "{$before}std::atexit({eventHandl
    ↪ er}); /\~*process-exit-handler~_{eventHandler}~*/",
    ↪ 20),
622 // Inside the scope replace:
623 // /\~*app-domain~_currentDomain~*/ ... _currentDomain.ProcessExit -= OnProcessExit;
624 // /\~*app-domain~_currentDomain~*/ ... /* No translation. It is not possible to
    ↪ unsubscribe from std::atexit. */
625 (new Regex(@"(?<before>(?<fieldScopeStart>/\~*app-domain~(?<field>[~\n\*]+)~\*/)(.|\n
    ↪ )+?)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),
626 // Inside the scope replace:
627 // /\~*process-exit-handler~OnProcessExit~*/ ... static void OnProcessExit(void
    ↪ *sender, EventArgs e)
628 // /\~*process-exit-handler~OnProcessExit~*/ ... static void OnProcessExit()
629 (new Regex(@"(?<before>(?<fieldScopeStart>/\~*process-exit-handler~(?<handler>[~\n\
    ↪ ]*)~\*/)(.|\n)+?static[\t ]+void[\t ]+k<handler>\(\^[~()\n]+\)\n)", "{$before}",
    ↪ 20),
630 // Remove scope borders.
631 // /\~*app-domain~_currentDomain~*/
632 //
633 (new Regex(@"/\~*[~\*\n]+(~[~\*\n]+)*~\*/"), "", 0),
634 // AppDomain.CurrentDomain.ProcessExit -= OnProcessExit;
635 // /* No translation. It is not possible to unsubscribe from std::atexit. */
636 (new Regex(@"AppDomain\.CurrentDomain\.ProcessExit -= ([a-zA-Z_][a-zA-Z0-9_]*)");
    ↪ "/* No translation. It is not possible to unsubscribe from std::atexit. */", 0),
637 }.Cast<ISubstitutionRule>().ToList();
638
639 /// <summary>
640 /// <para>
641 /// The to list.
642 /// </para>
643 /// <para></para>
644 /// </summary>
645 public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
646 {
647     // IDisposable disposable)
648     // IDisposable &disposable)
649     (new Regex(@"(?<argumentAbstractType>I[A-Z][a-zA-Z0-9]+(<[~>\r\n]+>)?)
    ↪ (?<argument>[_a-zA-Z0-9]+)(?<after>,|\))", "{$argumentAbstractType}
    ↪ &${argument}${after}", 0),
650     // ICounter<int, int> c1;
651     // ICounter<int, int>* c1;
652     (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[~>\r\n]+>)?)
    ↪ (?<variable>[_a-zA-Z0-9]+)(?<after> = null)?"); "{$abstractType}
    ↪ *${variable}${after}";, 0),
653     // (expression)
654     // expression
655     (new Regex(@"(\(|\)|)(([a-zA-Z0-9_]*:)+)\(|\)|;|\)|\n)"), "$1$2$3", 0),
656     // (method(expression))
657     // method(expression)

```



```

(new Regex(@"(?<firstSeparator>\(|\)|\((?<method>[a-zA-Z0-9_\->\*:]*)\)((?<expression>((?<parenthesis>\(|\)|(?<-parent
    ↪ hesis>)|[a-zA-Z0-9_\->\*:]*)+)(?(parenthesis)(?!))\)\)\)(?<lastSeparator>(,|
    ↪ |;|\\))"), "${firstSeparator}${method}(${expression})${lastSeparator}", 0),
659 // .append(".")
660 // .append(1, '.');
661 (new Regex(@"\.\append\""(^[^\\"]|\\\"")"\""), ".append(1, '$1')", 0),
662 // return ref _elements[node];
663 // return &_elements[node];
664 (new Regex(@"return ref ([_a-zA-Z0-9]+)\([([_a-zA-Z0-9*]+)\];"), "return &$1[$2];",
    ↪ 0),
665 // ((1, 2))
666 // ({1, 2})
667 (new Regex(@"(?<before>\(|,)\)((?<first>[^\n()]+),
    ↪ (?<second>[^\n()]+)\)((?<after>\)|,)", "${before}${{first},
    ↪ ${second}}${after}", 10),
668 // {1, 2}.GetHashCode()
669 // Platform::Hashing::Hash(1, 2)
670 (new Regex(@"{((?<first>[^\n{]+), (?<second>[^\n{]+))}\.GetHashCode\\(\)"),
    ↪ "Platform::Hashing::Hash(${first}, ${second})", 10),
671 // range.ToString()
672 // Platform::Converters::ToStd::string>(range).data()
673 (new Regex(@"(?<before>\W)(?<variable>[_a-zA-Z][_a-zA-Z0-9]+)\.ToString\\(\)"),
    ↪ "${before}Platform::Converters::ToStd::string>(${variable}).data()", 10),
674 // new
675 //
676 (new Regex(@"(?<before>\r?\n[^\r\n]*(\"\\\"|\"[^\r\n]*\"[^\r\n]*)(?<=\W)new\\
    ↪ s+\""), "${before}",
    ↪ 10),
677 // x == null
678 // x == nullptr
679 (new Regex(@"(?<before>\r?\n[^\r\n]*(\"\\\"|\"[^\r\n]*\"[^\r\n]*)(?<=\W)(?<v
    ↪ ariable>[_a-zA-Z][_a-zA-Z0-9]+)(?<operator>s*(==|!=)\s*)null(?<after>\W)"),
    ↪ "${before}${variable}${operator}nullptr${after}", 10),
680 // null
681 // {}
682 (new Regex(@"(?<before>\r?\n[^\r\n]*(\"\\\"|\"[^\r\n]*\"[^\r\n]*)(?<=\W)null\\
    ↪ (?<after>\W)"), "${before}-${after}",
    ↪ 10),
683 // default
684 // 0
685 (new Regex(@"(?<before>\r?\n[^\r\n]*(\"\\\"|\"[^\r\n]*\"[^\r\n]*)(?<=\W)defa
    ↪ ult(?<after>\W)"), "${before}0${after}",
    ↪ 10),
686 // object x
687 // void *x
688 (new Regex(@"(?<before>\r?\n[^\r\n]*(\"\\\"|\"[^\r\n]*\"[^\r\n]*)(?<=\W)(?<!
    ↪ @)(object|System\\.Object)(?<after>\w)"), "${before}void *${after}",
    ↪ 10),
689 // <object>
690 // <void*>
691 (new Regex(@"(?<before>\r?\n[^\r\n]*(\"\\\"|\"[^\r\n]*\"[^\r\n]*)(?<=\W)(?<!
    ↪ @)(object|System\\.Object)(?<after>\W)"), "${before}void*${after}",
    ↪ 10),
692 // @object
693 // object
694 (new Regex(@"@([_a-zA-Z0-9]+)"), "$1", 0),
695 // this->GetType().Name
696 // typeid(this).name()
697 (new Regex(@"(this)->GetType\\(\)\\.Name"), "typeid($1).name()", 0),
698 // ArgumentNullException
699 // std::invalid_argument
700 (new Regex(@"(?<before>\r?\n[^\r\n]*(\"\\\"|\"[^\r\n]*\"[^\r\n]*)(?<=\W)(Sys
    ↪ tem\\.)?ArgumentNullException(?<after>\W)"),
    ↪ "${before}std::invalid_argument${after}", 10),
701 // InvalidOperationException
702 // std::runtime_error
703 (new Regex(@"(\\W)(InvalidOperationException|Exception)(\\W)"),
    ↪ "$1std::runtime_error$3", 0),
704 // ArgumentException
705 // std::invalid_argument
706 (new Regex(@"(\\W)(ArgumentException|ArgumentOutOfRangeException)(\\W)"),
    ↪ "$1std::invalid_argument$3", 0),
707 // template <typename T> struct Range : IEquatable<Range<T>>
708 // template <typename T> struct Range {

```

```

709 (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),
710 // public: delegate void Disposal(bool manual, bool wasDisposed);
711 // public: delegate void Disposal(bool, bool);
712 (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),
713 // public: delegate void Disposal(bool, bool);
714 // using Disposal = void(bool, bool);
715 (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),
716 // <4-1>
717 // <3>
718 (new Regex(@"(?<before><)4-1(?(after>>)", "${before}3${after}", 0),
719 // <3-1>
720 // <2>
721 (new Regex(@"(?<before><)3-1(?(after>>)", "${before}2${after}", 0),
722 // <2-1>
723 // <1>
724 (new Regex(@"(?<before><)2-1(?(after>>)", "${before}1${after}", 0),
725 // <1-1>
726 // <0>
727 (new Regex(@"(?<before><)1-1(?(after>>)", "${before}0${after}", 0),
728 // #region Always
729 //
730 (new Regex(@"(^\r?\n)[ \t]*\#(region|endregion)[^\r\n]*(\r?\n|$)", "", 0),
731 // // #define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
732 //
733 (new Regex(@"\\\/[ \t]*\#define[ \t]+[_a-zA-Z0-9]+[ \t]*"), "", 0),
734 // #if USEARRAYPOOL\r\n#endif
735 //
736 (new Regex(@"#if [_a-zA-Z0-9]+\s+#endif", "", 0),
737 // [Fact]
738 //
739 (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),
740 // \A \n ... namespace
741 // \Anamespace
742 (new Regex(@"(\A)(\r?\n)+namespace"), "$1namespace", 0),
743 // \A \n ... class
744 // \Aclass
745 (new Regex(@"(\A)(\r?\n)+class"), "$1class", 0),
746 // \n\n\n
747 // \n\n
748 (new Regex(@"\r?\n[ \t]*\r?\n[ \t]*\r?\n"), Environment.NewLine +
    ↳ Environment.NewLine, 50),
749 // {\n\n
750 // {\n
751 (new Regex(@"{[ \t]*\r?\n[ \t]*\r?\n"), "{" + Environment.NewLine, 10),
752 // \n\n}
753 // \n}
754 (new Regex(@"\r?\n[ \t]*\r?\n(?<end>[ \t]*)"), Environment.NewLine + "${end}", 10),
755 }.Cast<ISubstitutionRule>().ToList();
756
757 /// <summary>
758 /// <para>
759 /// Initializes a new <see cref="CSharpToCppTransformer"/> instance.
760 /// </para>
761 /// <para></para>
762 /// </summary>
763 /// <param name="extraRules">
764 /// <para>A extra rules.</para>
765 /// <para></para>
766 /// </param>
767 public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
    ↳ base(FirstStage.Concat(extraRules).Concat>LastStage).ToList()) { }

```

```

768
769     /// <summary>
770     /// <para>
771     /// Initializes a new <see cref="CSharpToCppTransformer"/> instance.
772     /// </para>
773     /// <para></para>
774     /// </summary>
775     public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
776 }
777 }

```

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
22             class Program
23             {
24                 public static void Main(string[] args)
25                 {
26                     Console.WriteLine("Hello, world!");
27                 }
28             }";
29             const string expectedResult = @"class Program
30
31             public: static void Main(std::string args[])
32             {
33                 printf("Hello, world!\n");
34             }
35             };";
36
37             var transformer = new CSharpToCppTransformer();
38             var actualResult = transformer.Transform(helloWorldCode);
39             Assert.Equal(expectedResult, actualResult);
40         }
41     }
42 }

```

Index

./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs, 18
./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1