

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

```

```

58 //
59 (new Regex(@"\r?\n[^\n]+bool operator !=\(((?<type>[^\n]+) (?<left>[a-zA-Z0-9]+),
   ↳ \k<type> (?<right>[a-zA-Z0-9]+)\) => !\((\k<left>|\k<right>) ==
   ↳ (\k<left>|\k<right>)\);"), "", 10),
60 // public override bool Equals(object obj) => obj is Range<T> range ? Equals(range)
   ↳ : false;
61 //
62 (new Regex(@"\r?\n[^\n]+override bool Equals\((System\.)?[Oo]bject
   ↳ (?<this>[a-zA-Z0-9]+)\) => \k<this> is [^\n]+ (?<other>[a-zA-Z0-9]+) \?
   ↳ Equals\(\k<other>\) : false;"), "", 10),
63 // out TProduct
64 // TProduct
65 (new Regex(@"(?<before><|, ))(in|out)
   ↳ (?<typeParameter>[a-zA-Z0-9]+)(?<after>>|,))"),
   ↳ "${before}${typeParameter}${after}", 10),
66 // public ...
67 // public: ...
68 (new Regex(@"(?<newLineAndIndent>\r?\n?[
   ↳ \t]*) (?<before>[^\{\\(\r\n)*) (?<access>private|protected|public) [
   ↳ \t]+(?! [^\{\\(\r\n)*(interface|class|struct) [^\{\\(\r\n)*[\\(\r\n)]"),
   ↳ "${newLineAndIndent}${access}: ${before}", 0),
69 // public: static bool CollectExceptions { get; set; }
70 // public: inline static bool CollectExceptions;
71 (new Regex(@"(?<access>(private|protected|public): )(?<before>(static )?[^\r\n]+
   ↳ )(?<name>[a-zA-Z0-9]+) {[^;]}*(?<=\\W)get; [^;]}*(?<=\\W)set; [^;]}*"),
   ↳ "${access}inline ${before}${name};", 0),
72 // public abstract class
73 // class
74 (new Regex(@"((public|protected|private|internal|abstract|static)
   ↳ )*(?<category>interface|class|struct)", "${category}", 0),
75 //class GenericCollectionMethodsBase<TElement> {
76 // template <typename TElement> class GenericCollectionMethodsBase {
77 (new Regex(@"class ([a-zA-Z0-9]+)<([a-zA-Z0-9]+)>([^\{]+){", "template <typename $2>
   ↳ class $1$3{", 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 TProduct> class IFactory { public:
83 (new Regex(@"interface (?<interface>[a-zA-Z0-9]+)<(?!<typeParameters>[a-zA-Z0-9
   ↳ ,]+)>(?!<whitespace>[^\{]+){", "template <typename...> class ${interface};
   ↳ template <typename ${typeParameters}> class
   ↳ ${interface}<${typeParameters}>${whitespace}{ + Environment.NewLine + "
   ↳ public:", 0),
84 // template <typename TObject, TProperty, TValue>
85 // template <typename TObject, typename TProperty, TValue>
86 (new Regex(@"(?<before>template <((, )?typename [a-zA-Z0-9]+)+,
   ↳ )(?<typeParameter>[a-zA-Z0-9]+)(?<after>(,|>))"), "${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}${before}",
   ↳ 10),
93 // /*~extensionMethod~BuildExceptionString~*/...sb.BuildExceptionString(exception.Inn
   ↳ nerException, level +
   ↳ 1);
94 // /*~extensionMethod~BuildExceptionString~*/...BuildExceptionString(sb,
   ↳ exception.InnerException, level + 1);
95 (new Regex(@"(?<before>\/\*~extensionMethod~(?<name>[a-zA-Z0-9]+)~\*/(?:.|\\n)+\\W) (?<var
   ↳ iable>[_a-zA-Z0-9]+)\\. \k<name>\\(", "${before}${name}(${variable}, ",
   ↳ 50),
96 // Remove markers
97 // /*~extensionMethod~BuildExceptionString~*/
98 //

```

```

new Regex(@"/*~extensionMethod~[a-zA-Z0-9]+~*/"), "", 0),
// (this
// (
(new Regex(@"\((this ", "(", 0),
// public: static readonly EnsureAlwaysExtensionRoot Always = new
// EnsureAlwaysExtensionRoot();
// public: inline static EnsureAlwaysExtensionRoot Always;
(new Regex(@"(?<access>(private|protected|public): )?static readonly
    (?<type>[a-zA-Z0-9]+) (?<name>[a-zA-Z0-9_]+) = new \k<type>\(\);"),
    → "{access}inline static ${type} ${name};", 0),
// public: static readonly string ExceptionContentsSeparator = "---";
// public: inline static const char* ExceptionContentsSeparator = "---";
(new Regex(@"(?<access>(private|protected|public): )?(const|static readonly) string
    (?<name>[a-zA-Z0-9_]+) = \"\"(?<string>(\\"\"|\"\\r\\n\"))+\"\";"), "{access}inline
    → static const char* ${name} = \"${string}\";", 0),
// private: const int MaxPath = 92;
// private: inline static const int MaxPath = 92;
(new Regex(@"(?<access>(private|protected|public): )?(const|static readonly)
    (?<type>[a-zA-Z0-9]+) (?<name>[_a-zA-Z0-9]+) = (?<value>[~;\r\n]+);"),
    → "{access}inline static const ${type} ${name} = ${value};", 0),
// ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
// TArgument : class
// ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument* argument)
(new Regex(@"(?<before> [a-zA-Z]+\((([a-zA-Z *],)+, |)) (?<type>[a-zA-Z]+) (?<after>(\[
    → [a-zA-Z *,,]+\)) [ \r\n]+where \k<type> : class)", "{before}${type}*${after}",
    → 0),
// protected: abstract TElement GetFirst();
// protected: virtual TElement GetFirst() = 0;
(new Regex(@"(?<access>(private|protected|public): )?abstract
    (?<method>[~;\r\n]+);"), "{access}virtual ${method} = 0;", 0),
// TElement GetFirst();
// virtual TElement GetFirst() = 0;
(new Regex(@"([\r\n]+[ ]+)((?!return)[a-zA-Z0-9]+ [a-zA-Z0-9]+\([~\)\r\n]*\))([
    → ]*\[~\r\n]+)", "$1virtual $2 = 0$3", 1),
// protected: readonly TreeElement[] _elements;
// protected: TreeElement _elements[N];
(new Regex(@"(?<access>(private|protected|public): )?readonly
    (?<type>[a-zA-Z<0-9]+) ([\[]+ (?<name>[_a-zA-Z0-9]+);"), "{access}${type}
    → ${name}[N];", 0),
// protected: readonly TElement Zero;
// protected: TElement Zero;
(new Regex(@"(?<access>(private|protected|public): )?readonly
    (?<type>[a-zA-Z<0-9]+) (?<name>[_a-zA-Z0-9]+);"), "{access}${type} ${name};",
    → 0),
// internal
//
(new Regex(@"(\W)internal\s+)", "$1", 0),
// static void NotImplementedException(ThrowExtensionRoot root) => throw new
// NotImplementedException();
// static void NotImplementedException(ThrowExtensionRoot root) { return throw new
// NotImplementedException(); }
(new Regex(@"(^s+)(private|protected|public)?(: )?(template \<[~>\r\n]+\> )?(static
    → )?(override )?([a-zA-Z0-9]+
    → )([a-zA-Z0-9]+\)(([~\(\r\n]*)\)\s+=>\s+throw([~;\r\n]+);"),
    → "$1$2$3$4$5$6$7$8($9) { throw$10; }", 0),
// SizeBalancedTree(int capacity) => a = b;
// SizeBalancedTree(int capacity) { a = b; }
(new Regex(@"(^s+)(private|protected|public)?(: )?(template \<[~>\r\n]+\> )?(static
    → )?(override )?(void )?([a-zA-Z0-9]+)\)(([~\(\r\n]*)\)\s+=>\s+([~;\r\n]+);"),
    → "$1$2$3$4$5$6$7$8($9) { $10; }", 0),
// int SizeBalancedTree(int capacity) => a;
// int SizeBalancedTree(int capacity) { return a; }
(new Regex(@"(^s+)(private|protected|public)?(: )?(template \<[~>\r\n]+\> )?(static
    → )?(override )?([a-zA-Z0-9]+
    → )([a-zA-Z0-9]+\)(([~\(\r\n]*)\)\s+=>\s+([~;\r\n]+);"), "$1$2$3$4$5$6$7$8($9) {
    → return $10; }", 0),
// () => Integer<TElement>.Zero,
// () { return Integer<TElement>.Zero; },
(new Regex(@"\(\)\s+=>\s+(?<expression>[~() ,;\r\n]+\((((?<parenthesis>\()|(?<-parent
    → hesis>))|([~() ,;\r\n]*?)*)\))?[~() ,;\r\n]*)(?<after>,|\\);"), "() { return
    → ${expression}; }${after}", 0),
// => Integer<TElement>.Zero;
// { return Integer<TElement>.Zero; }
(new Regex(@"\)\s+=>\s+([~;\r\n]+?);"), ") { return $1; }", 0),
// () { return avlTree.Count; }
// [&]()-> auto { return avlTree.Count; }

```

```

147 (new Regex(@"(?<before>, |\\)\(\) { return (?<expression>[~;\r\n]+); }"),
148     ↳ "${before}[&]() -> auto { return ${expression}; }", 0),
149 // Count => GetSizeOrZero(Root);
150 // GetCount() { return GetSizeOrZero(Root); }
151 (new Regex(@"(\\W)([A-Z][a-zA-Z]+)\\s+=>\\s+([~;\r\n]+);"), "$1Get$2() { return $3; }",
152     ↳ 0),
153 // ArgumentInRange(const char* message) { const char* messageBuilder() { return
154     ↳ message; }
155 // ArgumentInRange(const char* message) { auto messageBuilder = [&]() -> const char*
156     ↳ { return message; };
157 (new Regex(@"(?<before>\\W[_a-zA-Z0-9]+\\(([^\\]\\n)*\\)[\\s\\n]*{([\\s\\n]*([^}]|\\n)*?(\\r?\\n)
158     ↳ ?[ \\t]*)(?<returnType>[_a-zA-Z0-9*:] +[_a-zA-Z0-9*:] *)
159     ↳ (?<methodName>[_a-zA-Z0-9]+)\\((?<arguments>[~\\]\\n)*\\)\\s*{(?<body>([~}]|\\n)+?)}"
160     ↳ ), "${before}auto ${methodName} = [&]() -> ${returnType} {${body}};",
161     ↳ 10),
162 // Func<TElement> treeCount
163 // std::function<TElement()> treeCount
164 (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<$1()> $2", 0),
165 // Action<TElement> free
166 // std::function<void(TElement)> free
167 (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<void($1)> $2",
168     ↳ 0),
169 // Predicate<TArgument> predicate
170 // std::function<bool(TArgument)> predicate
171 (new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<bool($1)>
172     ↳ $2", 0),
173 // var
174 // auto
175 (new Regex(@"(\\W)var(\\W)"), "$1auto$2", 0),
176 // unchecked
177 //
178 (new Regex(@"[\\r\\n]{2}\\s*?unchecked\\s*?$"), "", 0),
179 // throw new InvalidOperationException
180 // throw std::runtime_error
181 (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
182     ↳ std::runtime_error", 0),
183 // void RaiseExceptionIgnoredEvent(Exception exception)
184 // void RaiseExceptionIgnoredEvent(const std::exception& exception)
185 (new Regex(@"(\\(| ) (System\\.Exception|Exception) (|\\))"), "$1const
186     ↳ std::exception&$3", 0),
187 // EventHandler<Exception>
188 // EventHandler<std::exception>
189 (new Regex(@"(\\W) (System\\.Exception|Exception) (\\W)"), "$1std::exception$3", 0),
190 // override void PrintNode(TElement node, StringBuilder sb, int level)
191 // void PrintNode(TElement node, StringBuilder sb, int level) override
192 (new Regex(@"override ([a-zA-Z0-9 \\*\\+]+)\\(([^\\]\\r\\n|\\n)+?\\)"), "$1$2 override", 0),
193 // return (range.Minimum, range.Maximum)
194 // return {range.Minimum, range.Maximum}
195 (new Regex(@"(?<before>return\\s*)\\((?<values>[~\\]\\n)+\\)\\(?!\\() (?<after>\\W)"),
196     ↳ "${before}${values}}${after}", 0),
197 // string
198 // const char*
199 (new Regex(@"(\\W)string(\\W)"), "$1const char*$2", 0),
200 // System.ValueTuple
201 // std::tuple
202 (new Regex(@"(?<before>\\W) (System\\.)?ValueTuple(?:\\s*=) (?<after>\\W)"),
203     ↳ "${before}std::tuple${after}", 0),
204 // sbyte
205 // std::int8_t
206 (new Regex(@"(?<before>\\W) ((System\\.)?SB|sbyte(?:\\s*=) (?<after>\\W)"),
207     ↳ "${before}std::int8_t${after}", 0),
208 // short
209 // std::int16_t
210 (new Regex(@"(?<before>\\W) ((System\\.)?Int16|short(?:\\s*=) (?<after>\\W)"),
211     ↳ "${before}std::int16_t${after}", 0),
212 // int
213 // std::int32_t
214 (new Regex(@"(?<before>\\W) ((System\\.)?I|i)nt(32)?(?:\\s*=) (?<after>\\W)"),
215     ↳ "${before}std::int32_t${after}", 0),
216 // long
217 // std::int64_t
218 (new Regex(@"(?<before>\\W) ((System\\.)?Int64|long(?:\\s*=) (?<after>\\W)"),
219     ↳ "${before}std::int64_t${after}", 0),
220 // byte
221 // std::uint8_t

```

```

204 (new Regex(@"(?<before>\W)((System\.)?Byte|byte)(?!\\s*)(?<after>\W)",
    ↪ "${before}std::uint8_t${after}", 0),
205 // ushort
206 // std::uint16_t
207 (new Regex(@"(?<before>\W)((System\.)?UInt16|ushort)(?!\\s*)(?<after>\W)",
    ↪ "${before}std::uint16_t${after}", 0),
208 // uint
209 // std::uint32_t
210 (new Regex(@"(?<before>\W)((System\.)?UI|ui)nt(32)?(?!\\s*)(?<after>\W)",
    ↪ "${before}std::uint32_t${after}", 0),
211 // ulong
212 // std::uint64_t
213 (new Regex(@"(?<before>\W)((System\.)?UInt64|ulong)(?!\\s*)(?<after>\W)",
    ↪ "${before}std::uint64_t${after}", 0),
214 // char*[] args
215 // char* args[]
216 (new Regex(@"([_a-zA-Z0-9:\*]?)\\[\\] ([a-zA-Z0-9]+)", "$1 $2[]", 0),
217 // @object
218 // object
219 (new Regex(@"@([_a-zA-Z0-9]+)", "$1", 0),
220 // float.MinValue
221 // std::numeric_limits<float>::min()
222 (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+float|double)\\.MinValue(?<after>\W)",
    ↪ ")"), "${before}std::numeric_limits<${type}>::min()${after}",
    ↪ 0),
223 // double.MaxValue
224 // std::numeric_limits<float>::max()
225 (new Regex(@"(?<before>\W)(?<type>std::[a-z0-9_]+float|double)\\.MaxValue(?<after>\W)",
    ↪ ")"), "${before}std::numeric_limits<${type}>::max()${after}",
    ↪ 0),
226 // using Platform.Numbers;
227 //
228 (new Regex(@"([\\r\\n]{2}|^\\s*?using [\\a-zA-Z0-9]+;\\s*?$)", "", 0),
229 // struct TreeElement { }
230 // struct TreeElement { };
231 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\\s+){([\\sa-zA-Z0-9;:_]+?)}([~;])", "$1
    ↪ $2$3{$4};$5", 0),
232 // class Program { }
233 // class Program { };
234 (new Regex(@"(struct|class) ([a-zA-Z0-9]+[~\\r\\n]*)([\\r\\n]+(?<indentLevel>[\\t
    ↪ ]*))?\\{([\\S\\s]+?[\\r\\n]+\\k<indentLevel>)\\}([~;]|$)", "$1 $2$3{$4};$5", 0),
235 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
236 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
237 (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)", "class $1 : public $2", 0),
238 // class IProperty : ISetter<TValue, TObj>, IProvider<TValue, TObj>
239 // class IProperty : public ISetter<TValue, TObj>, IProvider<TValue, TObj>
240 (new Regex(@"(?<before>class [a-zA-Z0-9]+ : ((public [a-zA-Z0-9]+(<[a-zA-Z0-9
    ↪ ,]+>)?, )+)?(?<inheritedType>(?!public)[a-zA-Z0-9]+(<[a-zA-Z0-9
    ↪ ,]+>)?(?<after>(, [a-zA-Z0-9]+(?!>)|[ \\r\\n]+))", "${before}public
    ↪ ${inheritedType}${after}", 10),
241 // Insert scope borders.
242 // ref TElement root
243 // ~!root!~ref TElement root
244 (new Regex(@"(?<definition>(?!\\() (ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<ref>))
    ↪ (?<variable>[a-zA-Z0-9]+)(?=\\)|, | =))", "~!${variable}!~${definition}", 0),
245 // Inside the scope of ~!root!~ replace:
246 // root
247 // *root
248 (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),
249 // Remove scope borders.
250 // ~!root!~
251 //
252 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~", "", 5),
253 // ref auto root = ref
254 // ref auto root =
255 (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\\W)", "$1* $2 =$3", 0),
256 // *root = ref left;
257 // root = left;
258 (new Regex(@"\\*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\\W)", "$1 = $2$3", 0),
259 // (ref left)
260 // (left)
261 (new Regex(@"\\(ref ([a-zA-Z0-9]+)(\\)|\\(|,)", "($1$2", 0),
262 // ref TElement

```

```

263 // TElement*
264 (new Regex(@"( | \()ref ([a-zA-Z0-9]+) ", "$1$2* ", 0),
265 // ref sizeBalancedTree.Root
266 // &sizeBalancedTree->Root
267 (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)", "&$1->$2", 0),
268 // ref GetElement(node).Right
269 // &GetElement(node)->Right
270 (new Regex(@"ref ([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)",
    → "&$1($2)->$3", 0),
271 // GetElement(node).Right
272 // GetElement(node)->Right
273 (new Regex(@"([a-zA-Z0-9]+)\((([a-zA-Z0-9\*]+)\)\.([a-zA-Z0-9]+)", "$1($2)->$3", 0),
274 // [Fact]\npublic: static void SizeBalancedTreeMultipleAttachAndDetachTest()
275 // public: TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
276 (new Regex(@"\[Fact\]\[s\n\]+(public:)?(static)?void ([a-zA-Z0-9]+)\(\)", "public:
    → TEST_METHOD($3)", 0),
277 // class TreesTests
278 // TEST_CLASS(TreesTests)
279 (new Regex(@"class ([a-zA-Z0-9]+)Tests)", "TEST_CLASS($1)", 0),
280 // Assert.Equal
281 // Assert::AreEqual
282 (new Regex(@"(Assert)\.Equal)", "$1::AreEqual", 0),
283 // Assert.Throws
284 // Assert::ExpectException
285 (new Regex(@"(Assert)\.Throws)", "$1::ExpectException", 0),
286 // $"Argument {argumentName} is null."
287 // ((std::string)"Argument ").append(argumentName).append(" is null.").data()
288 (new Regex(@"\$" "(?<left>(\\" | [^"\\r\n])*" (?<expression>[_a-zA-Z0-9]+) (?<right>(\\"
    → \\" | [^"\\r\n])*" )",
    → "((std::string)$\" ${left}\").append(${expression}).append(\" ${right}\").data()",
    → 10),
289 // $"
290 // "
291 (new Regex(@"\$"""), "\"", 0),
292 // Console.WriteLine("...")
293 // printf("...\n")
294 (new Regex(@"Console\.WriteLine\(\"([~"\\r\n]+)\""\)", "printf(\"$1\\n\\n\"", 0),
295 // TElement Root;
296 // TElement Root = 0;
297 (new Regex(@"(\r?\n[\t ]+)(private|protected|public)?(:)?([a-zA-Z0-9]+)
    → )?([a-zA-Z0-9: ]+(?!return)) ([a-zA-Z0-9]+);", "$1$2$3$4 $5 = 0;", 0),
298 // TreeElement _elements[N];
299 // TreeElement _elements[N] = { {0} };
300 (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),
301 // auto path = new TElement[MaxPath];
302 // TElement path[MaxPath] = { {0} };
303 (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),
304 // bool Equals(Range<T> other) { ... }
305 // bool operator ==(const Key &other) const { ... }
306 (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),
307 // Insert scope borders.
308 // class Range { ... public: override const char* ToString() { return ...; }
309 // class Range { /*~Range~/ ... public: override const char* ToString() { return
    → ...; }
310 (new Regex(@"(?<classDeclarationBegin>\r?\n(?<indent>[\t ]*)(struct|class)
    → (?<type>[a-zA-Z0-9]+(<[^\n]+)?) [\t ]*(\r?\n)? [\t ]*(?<middle>((?!class|struc
    → t).|\n)+)? (?<toStringDeclaration>(?<access>(private|protected|public): )override
    → const char*\* ToString\(\)\)",
    → "${classDeclarationBegin}/*~${type}~/${middle}${toStringDeclaration}", 0),
311 // Inside the scope of ~!_exceptionsBag!~ replace:
312 // public: override const char* ToString() { return ...; }
313 // public: operator std::string() const { return ...; }\n\npublic: friend
    → std::ostream & operator << (std::ostream &out, const A &obj) { return out <<
    → (std::string)obj; }
314 (new Regex(@"(?<scope>/\*~(?<type>[_a-zA-Z0-9<>:]+)~\*/) (?<separator>.\|\\n) (?<before>
    → ((?!/\*~\k<type>~\*/)(.|\n))*? (?<toStringDeclaration>\r?\n(?<indent>[
    → \t]*) (?<access>(private|protected|public): )override const char*\* ToString\(\)
    → (?<toStringMethodBody>{[^\n]+}))",
    → "${scope}${separator}${before}${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),

```

```

315 // Remove scope borders.
316 // /*~Range~*/
317 //
318 (new Regex(@"/*~[_a-zA-Z0-9<>:]+\~*/"), "", 0),
319 // private: static readonly ConcurrentBag<std::exception> _exceptionsBag = new
    ↳ ConcurrentBag<std::exception>();
320 // private: inline static std::mutex _exceptionsBag_mutex; \n\n private: inline
    ↳ static std::vector<std::exception> _exceptionsBag;
321 (new Regex(@"(?<begin>\r?\n?(?<indent>[\t]+)) (?<access>(private|protected|public):
    ↳ )?static readonly ConcurrentBag<(?<argumentType>[~;\r\n]+)>
    ↳ (?<name>[_a-zA-Z0-9]+) = new ConcurrentBag<~k<argumentType>>\(\);"),
    ↳ "${begin}private: inline static std::mutex ${name}_mutex;" + Environment.NewLine
    ↳ + Environment.NewLine + "${indent}${access}inline static
    ↳ std::vector<${argumentType}> ${name};" , 0),
322 // public: static IReadOnlyCollection<std::exception> GetCollectedExceptions() {
    ↳ return _exceptionsBag; }
323 // public: static std::vector<std::exception> GetCollectedExceptions() { return
    ↳ std::vector<std::exception>(_exceptionsBag); }
324 (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}>(${fieldName}); }", 0),
325 // public: static event EventHandler<std::exception> ExceptionIgnored =
    ↳ OnExceptionIgnored; ... };
326 // ... public: static inline Platform::Delegates::MulticastDelegate<void(void*,
    ↳ const std::exception&> ExceptionIgnored = OnExceptionIgnored; };
327 (new Regex(@"(?<begin>\r?\n(\r?\n)?(?<halfIndent>[
    ↳ \t]+\k<halfIndent>)(?<access>(private|protected|public): )?static event
    ↳ EventHandler<(?<argumentType>[~;\r\n]+)> (?<name>[_a-zA-Z0-9]+) = (?<defaultDele_
    ↳ gate>[_a-zA-Z0-9]+);(?<middle>(\.|\n)+)?(?<end>\r?\n\k<halfIndent>});") ,
    ↳ "${middle}" + Environment.NewLine + Environment.NewLine +
    ↳ "${halfIndent}${halfIndent}${access}static inline
    ↳ Platform::Delegates::MulticastDelegate<void(void*, const ${argumentType}&>
    ↳ ${name} = ${defaultDelegate};${end}" , 0),
328 // Insert scope borders.
329 // class IgnoredExceptions { ... private: inline static std::vector<std::exception>
    ↳ _exceptionsBag;
330 // class IgnoredExceptions { /*~_exceptionsBag~*/ ... private: inline static
    ↳ std::vector<std::exception> _exceptionsBag;
331 (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),
332 // Inside the scope of ~!_exceptionsBag!~ replace:
333 // _exceptionsBag.Add(exception);
334 // _exceptionsBag.push_back(exception);
335 (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),
336 // Remove scope borders.
337 // /*~_exceptionsBag~*/
338 //
339 (new Regex(@"/*~[_a-zA-Z0-9]+\~*/"), "", 0),
340 // Insert scope borders.
341 // class IgnoredExceptions { ... private: static std::mutex _exceptionsBag_mutex;
342 // class IgnoredExceptions { /*~_exceptionsBag~*/ ... private: static std::mutex
    ↳ _exceptionsBag_mutex;
343 (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),
344 // Inside the scope of ~!_exceptionsBag!~ replace:
345 // return std::vector<std::exception>(_exceptionsBag);
346 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); return
    ↳ std::vector<std::exception>(_exceptionsBag);
347 (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),
348 // Inside the scope of ~!_exceptionsBag!~ replace:
349 // _exceptionsBag.Add(exception);
350 // std::lock_guard<std::mutex> guard(_exceptionsBag_mutex); \r\n
    ↳ _exceptionsBag.Add(exception);

```



```

351 (new Regex(@"(?<scope>/\~(?<fieldName>[_a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ e>((?!/\~\k<fieldName>~\*/)(.\|\\n))*?){(?<after>((?!lock_guard)([~{};]|\\n))*?\r
    ↪ ?\\n(?<indent>[ \t]*)\k<fieldName>[~;]\r\\n*;)"),
    ↪ "${scope}${separator}${before}{ " + Environment.NewLine +
    ↪ "${indent}std::lock_guard<std::mutex> guard(${fieldName}_mutex);${after}", 10),
352 // Remove scope borders.
353 // /*~_exceptionsBag~/
354 //
355 (new Regex(@"/*~[_a-zA-Z0-9]+~\*/"), "", 0),
356 // Insert scope borders.
357 // class IgnoredExceptions { ... public: static inline
    ↪ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
    ↪ ExceptionIgnored = OnExceptionIgnored;
358 // class IgnoredExceptions { /*~ExceptionIgnored~/ ... public: static inline
    ↪ Platform::Delegates::MulticastDelegate<void(void*, const std::exception&)>
    ↪ ExceptionIgnored = OnExceptionIgnored;
359 (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),
360 // Inside the scope of ~!ExceptionIgnored!~ replace:
361 // ExceptionIgnored.Invoke(NULL, exception);
362 // ExceptionIgnored(NULL, exception);
363 (new Regex(@"(?<scope>/\~(?<eventName>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ >((?!/\~\k<eventName>~\*/)(.\|\\n))*?)\k<eventName>\\.Invoke"),
    ↪ "${scope}${separator}${before}${eventName}", 10),
364 // Remove scope borders.
365 // /*~ExceptionIgnored~/
366 //
367 (new Regex(@"/*~[a-zA-Z0-9]+~\*/"), "", 0),
368 // Insert scope borders.
369 // auto added = new StringBuilder();
370 // /*~sb~/std::string added;
371 (new Regex(@"(auto|(System\\.Text\\.)?StringBuilder) (?<variable>[a-zA-Z0-9]+) = new
    ↪ (System\\.Text\\.)?StringBuilder\\(\\);"), "/*~${variable}~/std::string
    ↪ ${variable};", 0),
372 // static void Indent(StringBuilder sb, int level)
373 // static void Indent(/*~sb~/StringBuilder sb, int level)
374 (new Regex(@"(?<start>, \\() (System\\.Text\\.)?StringBuilder
    ↪ (?<variable>[a-zA-Z0-9]+)(?<end>, \\))"), "${start}/*~${variable}~/std::string&
    ↪ ${variable}${end}", 0),
375 // Inside the scope of ~!added!~ replace:
376 // sb.ToString()
377 // sb.data()
378 (new Regex(@"(?<scope>/\~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ ((?!/\~\k<variable>~\*/)(.\|\\n))*?)\k<variable>\\.ToString\\(\\)",
    ↪ "${scope}${separator}${before}${variable}.data()", 10),
379 // sb.AppendLine(argument)
380 // sb.append(argument).append('\\n')
381 (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(${argument}).append(1, '\\n')",
    ↪ 10),
382 // sb.Append('\\t', level);
383 // sb.append(level, '\\t');
384 (new Regex(@"(?<scope>/\~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ ((?!/\~\k<variable>~\*/)(.\|\\n))*?)\k<variable>\\.Append\\('(?<character>[~'\\r\\n]
    ↪ +)', (?<count>[~\\),\\r\\n]+)\\)"),
    ↪ "${scope}${separator}${before}${variable}.append(${count}, '${character}'))", 10),
385 // sb.AppendLine(argument)
386 // sb.append(argument)
387 (new Regex(@"(?<scope>/\~(?<variable>[a-zA-Z0-9]+)~\*/)(?<separator>.\|\\n)(?<before>
    ↪ ((?!/\~\k<variable>~\*/)(.\|\\n))*?)\k<variable>\\.Append\\((?!<argument>[~\\),\\r\\n]
    ↪ +)\\)"), "${scope}${separator}${before}${variable}.append(${argument})",
    ↪ 10),
388 // Remove scope borders.
389 // /*~sb~/
390 //
391 (new Regex(@"/*~[a-zA-Z0-9]+~\*/"), "", 0),
392 // Insert scope borders.
393 // auto added = new HashSet<TElement>();
394 // ~!added!~std::unordered_set<TElement> added;

```



```

(new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
    ↳ HashSet<(?<element>[a-zA-Z0-9]+)>\(\(\);",
    ↳ "~!${variable}!~std::unordered_set<${element}> ${variable};", 0),
// Inside the scope of ~!added!~ replace:
// added.Add(node)
// added.insert(node)
(new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.\|\\n)(?<before>((?<
    ↳ !~!\k<variable>!~)(.\|\\n))*?)\k<variable>\.Add\((?<argument>[a-zA-Z0-9]+)\)",
    ↳ "${scope}${separator}${before}${variable}.insert(${argument})", 10),
// Inside the scope of ~!added!~ replace:
// added.Remove(node)
// added.erase(node)
(new Regex(@"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>.\|\\n)(?<before>((?<
    ↳ !~!\k<variable>!~)(.\|\\n))*?)\k<variable>\.Remove\((?<argument>[a-zA-Z0-9]+)\)",
    ↳ "${scope}${separator}${before}${variable}.erase(${argument})", 10),
// if (added.insert(node)) {
// if (!added.contains(node)) { added.insert(node);
(new Regex(@"if \((?<variable>[a-zA-Z0-9]+)\.insert\((?<argument>[a-zA-Z0-9]+)\)\)(?
    ↳ <separator>[\t ]*[\r\\n]+)(?<indent>[\t ]*){", "if
    ↳ (!${variable}.contains(${argument})) ${separator}${indent}{ " +
    ↳ Environment.NewLine + "${indent}    ${variable}.insert(${argument});", 0),
// Remove scope borders.
// ~!added!~
//
(new Regex(@"~![a-zA-Z0-9]+!~"), "", 5),
// Insert scope borders.
// auto random = new System.Random();
// 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(@"(?<separator>(\|, |([\\W]) |return ))(?<!(->|\\*
    ↳ ))(?<method>(?!sizeof)[a-zA-Z0-9]+)\((?<!\) \{)",
    ↳ "${separator}this->${method}(", 1),
(new Regex(@"(?<scope> /\*method-start\*/ ) (?<before>((?<!\/*method-end\*/)(.\|\\n))*?) (
    ↳ ?<separator>[\\W] (?<!(?:|\\.|->)) (?<method>(?!sizeof)[a-zA-Z0-9]+)\((?<!\)
    ↳ \{) (?<after>.\|\\n)*?) (?<scopeEnd> /\*method-end\*/ )",
    ↳ "${scope}${before}${separator}this->${method} (${after}${scopeEnd}", 100),
// Remove scope borders.
// /*method-start*/
//
(new Regex(@" /\*method-(start|end)\*/"), "", 0),
// Insert scope borders.
// const std::exception& ex
// const std::exception& ex/~ex~/
(new Regex(@"(?<before>\\(| )(?<variableDefinition>(const )?(std::)?exception&
    ↳ (?<variable>[_a-zA-Z0-9]+)) (?<after>\\W)",
    ↳ "${before}${variableDefinition}/~${variable}~/ ${after}", 0),

```

```

// Inside the scope of ~!ex!~ replace:
// ex.Message
// ex.what()
(new Regex(@"(?<scope>/\~(?<variable>[_a-zA-Z0-9]+)\~*/)(?<separator>.\|\n)(?<before>
    >((?<!/!\~\k<variable>\~*/)(.\|\n))*?)\k<variable>\.Message"),
    > "${scope}${separator}${before}${variable}.what()", 10),
// Remove scope borders.
// /*~ex~*/
//
(new Regex(@"/*~[_a-zA-Z0-9]+\~*/"), "", 0),
// throw new ArgumentNullException(argumentName, message);
// throw std::invalid_argument(((std::string)"Argument
    > ").append(argumentName).append(" is null: ").append(message).append("."));
(new Regex(@"throw new
    > ArgumentNullException\(((?<argument>[_a-zA-Z]*[Aa]rgument[_a-zA-Z]*),
    > (?<message>[_a-zA-Z]*[Mm]essage[_a-zA-Z]*(\(\(\)?\)\);)", "throw
    > std::invalid_argument(((std::string)"Argument \").append(${argument}).append("\
    > is null: \").append(${message}).append("\.\\"));", 0),
// throw new ArgumentException(message, argumentName);
// throw std::invalid_argument(((std::string)"Invalid
    > ").append(argumentName).append(" argument: ").append(message).append("."));
(new Regex(@"throw new
    > ArgumentException\(((?<message>[_a-zA-Z]*[Mm]essage[_a-zA-Z]*(\(\(\)?\)\);", "throw
    > (?<argument>[_a-zA-Z]*[Aa]rgument[_a-zA-Z]*)\);", "throw
    > std::invalid_argument(((std::string)"Invalid \").append(${argument}).append("\
    > argument: \").append(${message}).append("\.\\"));", 0),
// throw new ArgumentOutOfRangeException(argumentName, argumentValue,
    > messageBuilder());
// throw std::invalid_argument(((std::string)"Value
    > [").append(std::to_string(argumentValue)).append("] of argument
    > [").append(argumentName).append("] is out of range:
    > ").append(messageBuilder()).append("."));
(new Regex(@"throw new ArgumentOutOfRangeException\(((?<argument>[_a-zA-Z]*[Aa]rgument
    > [_a-zA-Z]*([Nn]ame[_a-zA-Z]*)?),
    > (?<argumentValue>[_a-zA-Z]*[Aa]rgument[_a-zA-Z]*([Vv]alue[_a-zA-Z]*)?),
    > (?<message>[_a-zA-Z]*[Mm]essage[_a-zA-Z]*(\(\(\)?\)\);", "throw
    > std::invalid_argument(((std::string)"Value
    > [").append(std::to_string(${argumentValue})).append("] of argument
    > [").append(${argument}).append("] is out of range:
    > \").append(${message}).append("\.\\"));", 0),
// throw new NotSupportedException();
// throw std::logic_error("Not supported exception.");
(new Regex(@"throw new NotSupportedException\(\(\);", "throw std::logic_error(\"Not
    > supported exception.\");", 0),
// throw new NotImplementedException();
// throw std::logic_error("Not implemented exception.");
(new Regex(@"throw new NotImplementedException\(\(\);", "throw std::logic_error(\"Not
    > implemented exception.\");", 0),
}.Cast<ISubstitutionRule>().ToList();

public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
{
    // ICounter<int, int> c1;
    // ICounter<int, int>* c1;
    (new Regex(@"(?<abstractType>I[A-Z][_a-zA-Z0-9]+(<[~>\r\n]+)?>
        > (?<variable>[_a-zA-Z0-9]+);", "${abstractType}* ${variable};", 0),
    // (expression)
    // expression
    (new Regex(@"(\(|\)|\((([_a-zA-Z0-9_!~*:]*)\)|(|\;|\)|\)))", "$1$2$3", 0),
    // (method(expression))
    // method(expression)
    (new Regex(@"(?<firstSeparator>(\(|
        > ))\(((?<method>[_a-zA-Z0-9_!~*:]*)\)|(((?<expression>((?<parenthesis>\(|(?<-parent
        > hesis>)|[_a-zA-Z0-9_!~*:]*)\)|(?<parenthesis>(?!))\)|(?<lastSeparator>(|
        > |;|\)|\)))", "${firstSeparator}${method}(${expression})${lastSeparator}", 0),
    // return ref _elements[node];
    // return &_elements[node];
    (new Regex(@"return ref ([_a-zA-Z0-9_!~*:]*)\&([[_a-zA-Z0-9_!~*:]*)\];", "return &$1[$2];",
        > 0),
    // null
    // nullptr
    (new Regex(@"(?<before>\r?\n[~""\r\n]*(""\|""|~""\r\n)*""[~""\r\n]*)?(?<=\\W)null
        > (?<after>\\W)", "${before}nullptr${after}",
        > 10),
    // default

```

```

487 // 0
488 (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"|~""\r\n))*""[~""\r\n]*)(?<=\\W)defa
  ↳ ult(?<after>\\W)"), "${before}0${after}",
  ↳ 10),
489 // object x
490 // void *x
491 (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"|~""\r\n))*""[~""\r\n]*)(?<=\\W)([O|
  ↳ o]bject|System\\.Object) (?<after>\\W)"), "${before}void *${after}",
  ↳ 10),
492 // <object>
493 // <void*>
494 (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"|~""\r\n))*""[~""\r\n]*)(?<=\\W)(?!
  ↳ \\W)([O|o]bject|System\\.Object) (?<after>\\W)"), "${before}void*${after}",
  ↳ 10),
495 // ArgumentNullException
496 // std::invalid_argument
497 (new Regex(@"(?<before>\r?\n[~""\r\n]*("(\\"|~""\r\n))*""[~""\r\n]*)(?<=\\W)(Sys
  ↳ tem\\.)?ArgumentNullException(?<after>\\W)"),
  ↳ "${before}std::invalid_argument${after}", 10),
498 // struct Range<T> : IEquatable<Range<T>> {
499 // struct Range<T> {
500 (new Regex(@"(?<before>(struct|class) (?<type>[a-zA-Z0-9]+(<[~\\n]+)?)) :
  ↳ IEquatable<\\k<type>>(?<after>(\\s|\\n)*{)"), "${before}${after}", 0),
501 // #region Always
502 //
503 (new Regex(@"(~|\\r?\\n)[ \\t]*#(region|endregion)[~\\r\\n]*(\\r?\\n|$)"), "", 0),
504 // //define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
505 //
506 (new Regex(@"\\/\\/[/ \\t]*#define[ \\t]+[_a-zA-Z0-9]+[ \\t]*"), "", 0),
507 // #if USEARRAYPOOL\\r\\n#endif
508 //
509 (new Regex(@"#if [a-zA-Z0-9]+\\s+#endif"), "", 0),
510 // [Fact]
511 //
512 (new Regex(@"(?<firstNewLine>\\r?\\n|\\A)(?<indent>[\\t
  ↳ ]+)[\\[a-zA-Z0-9]+(\\((?<expression>(\\(?<parenthesis>\\(|(?<-parenthesis>\\))|(~|\\r
  ↳ \\n)*+)(?<parenthesis>(?!))\\))]?[ \\t]*(\\r?\\n\\k<indent>)?"),
  ↳ "${firstNewLine}${indent}", 5),
513 // \\n ... namespace
514 // namespace
515 (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+namespace"), "$1namespace", 0),
516 // \\n ... class
517 // class
518 (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+class"), "$1class", 0),
519 // \\n\\n\\n
520 // \\n\\n
521 (new Regex(@"\\r?\\n[ \\t]*\\r?\\n[ \\t]*\\r?\\n"), Environment.NewLine +
  ↳ Environment.NewLine, 50),
522 // {\\n\\n
523 // {\\n
524 (new Regex(@"{[ \\t]*\\r?\\n[ \\t]*\\r?\\n"), "{" + Environment.NewLine, 10),
525 // \\n\\n}
526 // {\\n
527 (new Regex(@"\\r?\\n[ \\t]*\\r?\\n(?<end>[ \\t]*}")), Environment.NewLine + "${end}", 10),
528 }.Cast<ISubstitutionRule>().ToList();
529
530 public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
  ↳ base(FirstStage.Concat(extraRules).Concat>LastStage).ToList()) { }
531
532 public CSharpToCppTransformer() : base(FirstStage.Concat>LastStage).ToList()) { }
533 }
534 }

```

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("");

```

```
13         Assert.Equal("", actualResult);
14     }
15
16     [Fact]
17     public void HelloWorldTest()
18     {
19         const string helloWorldCode = @"using System;
20 class Program
21 {
22     public static void Main(string[] args)
23     {
24         Console.WriteLine(""Hello, world!"");
25     }
26 }";
27         const string expectedResult = @"class Program
28 {
29     public: static void Main(const char* args[])
30     {
31         printf(""Hello, world!\n"");
32     }
33 };";
34         var transformer = new CSharpToCppTransformer();
35         var actualResult = transformer.Transform(helloWorldCode);
36         Assert.Equal(expectedResult, actualResult);
37     }
38 }
39 }
```

Index

./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs, 11
./csharp/Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1