

1.1 ./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 : Transformer
11     {
12         public static readonly IList

```

```

56 // private const int MaxPath = 92;
57 // static const int MaxPath = 92;
58 (new Regex(@"private (const|static readonly) ([a-zA-Z0-9]+) ([_a-zA-Z0-9]+) =
→ ([~;\r\n]+);"), "static const $2 $3 = $4;", null, 0),
59 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument argument) where
→ TArgument : class
60 // ArgumentNotNull(EnsureAlwaysExtensionRoot root, TArgument& argument)
61 (new Regex(@"(?<before> [a-zA-Z]+\\((([a-zA-Z *],+ |)) (?<type>[a-zA-Z]+) (?<after>([
→ [a-zA-Z *],+))\\)) [ \r\n]+where \k<type> : class)", "${before}${type}&${after}",
→ null, 0),
62 // protected virtual
63 // virtual
64 (new Regex(@"protected virtual"), "virtual", null, 0),
65 // protected abstract TElement GetFirst();
66 // virtual TElement GetFirst() = 0;
67 (new Regex(@"protected abstract ([~;\r\n]+);"), "virtual $1 = 0;", null, 0),
68 // TElement GetFirst();
69 // virtual TElement GetFirst() = 0;
70 (new Regex(@"([ \r\n]+[ ]+)((?!return)[a-zA-Z0-9]+ [a-zA-Z0-9]+\\([~\r\n]*))"; [
→ ]*\\r\n]+)", "$1virtual $2 = 0$3", null, 1),
71 // public virtual
72 // virtual
73 (new Regex(@"public virtual"), "virtual", null, 0),
74 // protected readonly
75 //
76 (new Regex(@"protected readonly "), "", null, 0),
77 // protected readonly TreeElement[] _elements;
78 // TreeElement _elements[N];
79 (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+)([\\[\\]]+)
→ ([_a-zA-Z0-9]+);"), "$2 $4[N];", null, 0),
80 // protected readonly TElement Zero;
81 // TElement Zero;
82 (new Regex(@"(protected|private) readonly ([a-zA-Z<>0-9]+) ([_a-zA-Z0-9]+);"), "$2
→ $3;", null, 0),
83 // private
84 //
85 (new Regex(@"(\\W)(private|protected|public|internal) "), "$1", null, 0),
86 // static void NotImplementedException(ThrowExtensionRoot root) => throw new
→ NotImplementedException();
87 // static void NotImplementedException(ThrowExtensionRoot root) { return throw new
→ NotImplementedException(); }
88 (new Regex(@"(^\\s+)(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) {
→ throw$8; }", null, 0),
89 // SizeBalancedTree(int capacity) => a = b;
90 // SizeBalancedTree(int capacity) { a = b; }
91 (new Regex(@"(^\\s+)(template \\<[~>\\r\\n]+> )?(static )?(override )?(void
→ )?([a-zA-Z0-9]+)\\((([~\r\n]*))\\s+=>\\s+([~;\r\n]+);"), "$1$2$3$4$5$6($7) { $8;
→ }", null, 0),
92 // int SizeBalancedTree(int capacity) => a;
93 // int SizeBalancedTree(int capacity) { return a; }
94 (new Regex(@"(^\\s+)(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) {
→ return $8; }", null, 0),
95 // () => Integer<TElement>.Zero,
96 // () { return Integer<TElement>.Zero; },
97 (new Regex(@"\\(\\)\\s+=>\\s+([~;\r\n]+?);"), "() { return $1; }", null, 0),
98 // => Integer<TElement>.Zero;
99 // { return Integer<TElement>.Zero; }
100 (new Regex(@"\\)\\s+=>\\s+([~;\r\n]+?);"), ") { return $1; }", null, 0),
101 // () { return avlTree.Count; }
102 // [&]()-> auto { return avlTree.Count; }
103 (new Regex(@"\\(\\) { return ([~;\r\n]+); }"), "", [&]()-> auto { return $1; }",
→ null, 0),
104 // Count => GetSizeOrZero(Root);
105 // GetCount() { return GetSizeOrZero(Root); }
106 (new Regex(@"(\\W)([A-Z][a-zA-Z]+)\\s+=>\\s+([~;\r\n]+);"), "$1Get$2() { return $3; }",
→ null, 0),
107 // Func<TElement> treeCount
108 // std::function<TElement()> treeCount
109 (new Regex(@"Func<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)", "std::function<$1()> $2", null,
→ 0),
110 // Action<TElement> free
111 // std::function<void(TElement)> free
112 (new Regex(@"Action<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)", "std::function<void($1)> $2",
→ null, 0),

```

```

113 // Predicate<TArgument> predicate
114 // std::function<bool(TArgument)> predicate
115 (new Regex(@"Predicate<([a-zA-Z0-9]+)> ([a-zA-Z0-9]+)"), "std::function<bool($1)>
    → $2", null, 0),
116 // var
117 // auto
118 (new Regex(@"(\W)var(\W)"), "$1auto$2", null, 0),
119 // unchecked
120 //
121 (new Regex(@"[\r\n]{2}\s*unchecked\s*?$"), "", null, 0),
122 // $"
123 // "
124 (new Regex(@"\"$\""), "\"", null, 0),
125 // Console.WriteLine(...)
126 // printf(...)
127 (new Regex(@"Console.WriteLine\(\"([^\r\n]+)\"\)"), "printf(\"$1\\n\")", null, 0),
128 // throw new InvalidOperationException
129 // throw std::runtime_error
130 (new Regex(@"throw new (InvalidOperationException|Exception)"), "throw
    → std::runtime_error", null, 0),
131 // void RaiseExceptionIgnoredEvent(Exception exception)
132 // void RaiseExceptionIgnoredEvent(const std::exception& exception)
133 (new Regex(@"(\(|\|)(System\.Exception|Exception)(\|)\""), "$1const
    → std::exception&$3", null, 0),
134 // EventHandler<Exception>
135 // EventHandler<std::exception>
136 (new Regex(@"(\W)(System\.Exception|Exception)(\W)"), "$1std::exception$3", null, 0),
137 // override void PrintNode(TElement node, StringBuilder sb, int level)
138 // void PrintNode(TElement node, StringBuilder sb, int level) override
139 (new Regex(@"override ([a-zA-Z0-9 \*+]+)(\([^\\r\n]+?\))"), "$1$2 override", null,
    → 0),
140 // string
141 // char*
142 (new Regex(@"(\W)string(\W)"), "$1char*$2", null, 0),
143 // sbyte
144 // std::int8_t
145 (new Regex(@"(\W)sbyte(\W)"), "$1std::int8_t$2", null, 0),
146 // uint
147 // std::uint32_t
148 (new Regex(@"(\W)uint(\W)"), "$1std::uint32_t$2", null, 0),
149 // char*[] args
150 // char* args[]
151 (new Regex(@"([_a-zA-Z0-9:\*]?)\[\] ([a-zA-Z0-9]+)"), "$1 $2[]", null, 0),
152 // @object
153 // object
154 (new Regex(@"@([_a-zA-Z0-9]+)"), "$1", null, 0),
155 // using Platform.Numbers;
156 //
157 (new Regex(@"([\r\n]{2}|~)\s*?using [\.a-zA-Z0-9]+;\s*?$"), "", null, 0),
158 // struct TreeElement { }
159 // struct TreeElement { };
160 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([\^;])"), "$1
    → $2$3{$4};$5", null, 0),
161 // class Program { }
162 // class Program { };
163 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)[^\r\n]*([\r\n]+(?<indentLevel>[\t
    → ]*)?){([\S\s]+?[\r\n]+\k<indentLevel>)}([\^;]|$)"), "$1 $2$3{$4};$5", null, 0),
164 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
165 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
166 (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)"), "class $1 : public $2", null,
    → 0),
167 // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
168 // class IProperty : public ISetter<TValue, TObject>, IProvider<TValue, TObject>
169 (new Regex(@"(?<before>class [a-zA-Z0-9]+ : ((public [a-zA-Z0-9]+(<[a-zA-Z0-9
    → ,]+)?), +)?)(?<inheritedType>(?!public) [a-zA-Z0-9]+(<[a-zA-Z0-9
    → ,]+)?)(?<after>(, [a-zA-Z0-9]+(?!>)|[\r\n]+))"), "${before}public
    → ${inheritedType}${after}", null, 10),
170 // Insert scope borders.
171 // ref TElement root
172 // ~!root!~ref TElement root
173 (new Regex(@"(?<definition>(?!=|\\() (ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<!ref))
    → (?<variable>[a-zA-Z0-9]+)(?=\\|,|=)"), "~!${variable}!~${definition}", null,
    → 0),
174 // Inside the scope of ~!root!~ replace:
175 // root
176 // *root

```

```

177 (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}", null, 70),
178 // Remove scope borders.
179 // ~!root!~
180 //
181 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
182 // ref auto root = ref
183 // ref auto root =
184 (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)", "$1* $2 =$3", null, 0),
185 // *root = ref left;
186 // root = left;
187 (new Regex(@"\*( [a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)", "$1 = $2$3", null, 0),
188 // (ref left)
189 // (left)
190 (new Regex(@"\(\ref ([a-zA-Z0-9]+)(\)|\(|,)", "($1$2", null, 0),
191 // ref TElement
192 // TElement*
193 (new Regex(@"( |\\())ref ([a-zA-Z0-9]+) "), "$1$2* ", null, 0),
194 // ref sizeBalancedTree.Root
195 // &sizeBalancedTree->Root
196 (new Regex(@"ref ([a-zA-Z0-9]+)\.([a-zA-Z0-9\*]+)", "&$1->$2", null, 0),
197 // ref GetElement(node).Right
198 // &GetElement(node)->Right
199 (new Regex(@"ref ([a-zA-Z0-9]+)\((( [a-zA-Z0-9\*]+)\)\)\.([a-zA-Z0-9]+)",
    ↳ "&$1($2)->$3", null, 0),
200 // GetElement(node).Right
201 // GetElement(node)->Right
202 (new Regex(@"([a-zA-Z0-9]+)\((( [a-zA-Z0-9\*]+)\)\)\.([a-zA-Z0-9]+)", "$1($2)->$3",
    ↳ null, 0),
203 // [Fact]\npublic static void SizeBalancedTreeMultipleAttachAndDetachTest()
204 // TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
205 (new Regex(@"[Fact\\][\s\n]+(static)?void ([a-zA-Z0-9]+)\(\n)", "TEST_METHOD($2)",
    ↳ null, 0),
206 // class TreesTests
207 // TEST_CLASS(TreesTests)
208 (new Regex(@"class ([a-zA-Z0-9]+)Tests", "TEST_CLASS($1)", null, 0),
209 // Assert.Equal
210 // Assert::AreEqual
211 (new Regex(@"Assert\.Equal"), "Assert::AreEqual", null, 0),
212 // TEElement Root;
213 // TEElement Root = 0;
214 (new Regex(@"(\r?\n[\t ]+)([a-zA-Z0-9:_]+(?!return)) ([_a-zA-Z0-9]+);"), "$1$2 $3 =
    ↳ 0;", null, 0),
215 // TreeElement _elements[N];
216 // TreeElement _elements[N] = { {0} };
217 (new Regex(@"(\r?\n[\t ]+)([a-zA-Z0-9]+) ([_a-zA-Z0-9]+)\((( [a-zA-Z0-9]+)\);",
    ↳ "$1$2 $3[$4] = { {0} };", null, 0),
218 // auto path = new TEElement[MaxPath];
219 // TEElement path[MaxPath] = { {0} };
220 (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} };", null, 0),
221 // Insert scope borders.
222 // auto added = new HashSet<TElement>();
223 // ~!added!~std::unordered_set<TElement> added;
224 (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
    ↳ HashSet<(?<element>[a-zA-Z0-9]+)>\(\n);"),
    ↳ "~!${variable}!~std::unordered_set<${element}> ${variable};", null, 0),
225 // Inside the scope of ~!added!~ replace:
226 // added.Add(node)
227 // added.insert(node)
228 (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})", null, 10),
229 // Inside the scope of ~!added!~ replace:
230 // added.Remove(node)
231 // added.erase(node)
232 (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})", null, 10),
233 // if (added.insert(node)) {
234 // if (!added.contains(node)) { added.insert(node);

```

```

235 (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});", null, 0),
236 // Remove scope borders.
237 // ~!added!~
238 //
239 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
240 // Insert scope borders.
241 // auto random = new System.Random();
242 // std::srand(0);
243 (new Regex(@"[a-zA-Z0-9\.]+ ([a-zA-Z0-9]+) = new
    ↳ (System\.)?Random\((([a-zA-Z0-9]+)\)\);", "~!$1!~std::srand($3);", null, 0),
244 // Inside the scope of ~!random!~ replace:
245 // random.Next(1, N)
246 // (std::rand() % N) + 1
247 (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}", null, 10),
248 // Remove scope borders.
249 // ~!random!~
250 //
251 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
252 // Insert method body scope starts.
253 // void PrintNodes(TElement node, StringBuilder sb, int level) {
254 // void PrintNodes(TElement node, StringBuilder sb, int level) {/*method-start*/
255 (new Regex(@"(?<start>\r?\n[\t ]+)(?<prefix>((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*/${end}", null,
    ↳ 0),
256 // Insert method body scope ends.
257 // {/*method-start*/...}
258 // {/*method-start*/.../*method-end*/}
259 (new Regex(@"{/{/*method-start*//(?<body>((?<bracket>\{)|(?<-bracket>\})|[^{\}]*)+)
    ↳ \}"}", "{/{/*method-start*/${body}/*method-end*/}", null,
    ↳ 0),
260 // Inside method bodies replace:
261 // GetFirst(
262 // this->GetFirst(
263 // (new Regex(@"(?<separator>(\(|\)|([\W])|return ))(?<!(->|\*
    ↳ ))(?<method>(?!sizeof)[a-zA-Z0-9]+\(((?!\\) \{)\"",
    ↳ "${separator}this->${method}(", null, 1),
264 (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}", null, 100),
265 // Remove scope borders.
266 // /*method-start*/
267 //
268 (new Regex(@"/{/*method-(start|end)*/}"), "", null, 0),
269 // throw new ArgumentNullException(argumentName, message);
270 // throw std::invalid_argument(((std::string)"Argument
    ↳ ").append(argumentName).append(" is null: ").append(message).append("."));
271 (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("\.\\"));", null, 0),
272 // throw new ArgumentException(message, argumentName);
273 // throw std::invalid_argument(((std::string)"Invalid
    ↳ ").append(argumentName).append(" argument: ").append(message).append("."));
274 (new Regex(@"throw new ArgumentException\(((?<message>[a-zA-Z]*[Mm]essage[a-zA-Z]*),
    ↳ (?<argument>[a-zA-Z]*[Aa]rgument[a-zA-Z]*\)\);", "throw
    ↳ std::invalid_argument(((std::string)"Invalid \").append(${argument}).append("\
    ↳ argument: \").append(${message}).append("\.\\"));", null, 0),
275 // throw new NotSupportedException();
276 // throw std::logic_error("Not supported exception.");
277 (new Regex(@"throw new NotSupportedException\(\);", "throw std::logic_error(\"Not
    ↳ supported exception.\");", null, 0),
278 // throw new NotImplementedException();
279 // throw std::logic_error("Not implemented exception.");
280 (new Regex(@"throw new NotImplementedException\(\);", "throw std::logic_error(\"Not
    ↳ implemented exception.\");", null, 0),
281

```

```

282     }.Cast<ISubstitutionRule>().ToList();
283
284     public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
285     {
286         // ICounter<int, int> c1;
287         // ICounter<int, int>* c1;
288         (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[^\r\n]+>)?)
289         ↪ (?<variable>[_a-zA-Z0-9]+);"), "${abstractType}* ${variable}";, null, 0),
289         // (expression)
290         // expression
291         (new Regex(@"(\(|\)|)(([a-zA-Z0-9_\*:]+)\(|\)|;|\\|)"), "$1$2$3", null, 0),
292         // (method(expression))
293         // method(expression)
294         (new Regex(@"(?<firstSeparator>(\(|
295         ↪ ))\(((?<method>[a-zA-Z0-9_\*:]+)\(((?<expression>((?<parenthesis>\(|(?<-parent
296         ↪ hesis>))|([a-zA-Z0-9_\*:]+)(?(parenthesis)(?!))\)|(?<lastSeparator>(|
297         ↪ |;|\\|)))"), "${firstSeparator}${method}(${expression})${lastSeparator}";, null, 0),
298         // return ref _elements[node];
299         // return &_elements[node];
300         (new Regex(@"return ref ([_a-zA-Z0-9]+)\([([_a-zA-Z0-9_\*:]+)\];"), "return &$1[$2]";,
301         ↪ null, 0),
302         // default
303         // 0
304         (new Regex(@"(\W)default(\W)"), "${1}0$2", null, 0),
305         // // #define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
306         //
307         (new Regex(@"\\\/[ \t]*#define[ \t]+[_a-zA-Z0-9]+[ \t]*"), "", null, 0),
308         // #if USEARRAYPOOL\r\n#endif
309         //
310         (new Regex(@"#if [a-zA-Z0-9]+\s+#endif"), "", null, 0),
311         // [Fact]
312         //
313         (new Regex(@"(?<firstNewLine>\r?\n|\A)(?<indent>[ \t
314         ↪ ]+)\[[a-zA-Z0-9]+(\(((?<expression>((?<parenthesis>\(|(?<-parenthesis>))|(?<-parenthesis>))|(?<-parenthesis>))|(?<-parenthesis>))\]|([^\r\n
315         ↪ \n]*)+(?(parenthesis)(?!))\))?\][ \t]*(\r?\n\k<indent>)?"),
316         ↪ "${firstNewLine}${indent}";, null, 5),
317         // \n ... namespace
318         // namespace
319         (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+namespace"), "$1namespace", null, 0),
320         // \n ... class
321         // class
322         (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+class"), "$1class", null, 0),
323     }.Cast<ISubstitutionRule>().ToList();
324
325     public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
326     ↪ base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
327
328     public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
329 }
330 }

```

1.2 ./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 HelloWorldTest()
9          {
10             const string helloWorldCode = @"using System;
11 class Program
12 {
13     public static void Main(string[] args)
14     {
15         Console.WriteLine(""Hello, world!"");
16     }
17 }";
18             const string expectedResult = @"class Program
19 {
20     public:
21     static void Main(char* args[])
22     {
23         printf(""Hello, world!\n"");
24     }
25 };";
26             var transformer = new CSharpToCppTransformer();

```

```
27         var actualResult = transformer.Transform(helloWorldCode, new Context(null));
28         Assert.Equal(expectedResult, actualResult);
29     }
30 }
31 }
```

Index

./Platform.RegularExpressions.Transformer.CSharpToCpp.Tests/CSharpToCppTransformerTests.cs, 6
./Platform.RegularExpressions.Transformer.CSharpToCpp/CSharpToCppTransformer.cs, 1