

LinksPlatform's Platform.RegularExpressions.Transformer.CSharpToCpp Class Library

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

```

```

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

```

```

117 //
118 (new Regex(@"[\r\n]{2}\s*unchecked\s*?$"), "", null, 0),
119 // $"
120 // "
121 (new Regex(@"\$"""), "\"", null, 0),
122 // Console.WriteLine(...)
123 // printf(...)
124 (new Regex(@"Console.WriteLine\(\"([^\r\n]+)\""\), "printf(\"$1\\n\")", null, 0),
125 // throw new InvalidOperationException
126 // throw std::runtime_error
127 (new Regex(@"throw new (InvalidOperationException|Exception)", "throw
→ std::runtime_error", null, 0),
128 // void RaiseExceptionIgnoredEvent(Exception exception)
129 // void RaiseExceptionIgnoredEvent(const std::exception& exception)
130 (new Regex(@"(\(|\|)(System\.Exception|Exception)(\|)")), "$1const
→ std::exception&$3", null, 0),
131 // EventHandler<Exception>
132 // EventHandler<std::exception>
133 (new Regex(@"(\W)(System\.Exception|Exception)(\W)"), "$1std::exception$3", null, 0),
134 // override void PrintNode(TElement node, StringBuilder sb, int level)
135 // void PrintNode(TElement node, StringBuilder sb, int level) override
136 (new Regex(@"override ([a-zA-Z0-9 \*+])\(\([^\\r\n]+?\)\)"), "$1$2 override", null,
→ 0),
137 // string
138 // char*
139 (new Regex(@"(\W)string(\W)"), "$1char*$2", null, 0),
140 // sbyte
141 // std::int8_t
142 (new Regex(@"(\W)sbyte(\W)"), "$1std::int8_t$2", null, 0),
143 // uint
144 // std::uint32_t
145 (new Regex(@"(\W)uint(\W)"), "$1std::uint32_t$2", null, 0),
146 // char*[] args
147 // char* args[]
148 (new Regex(@"([_a-zA-Z0-9:*\?]|\[ \] ([a-zA-Z0-9]+)"), "$1 $2[]", null, 0),
149 // @object
150 // object
151 (new Regex(@"@([_a-zA-Z0-9]+)"), "$1", null, 0),
152 // using Platform.Numbers;
153 //
154 (new Regex(@"([\r\n]{2}|~)\s*?using [\.a-zA-Z0-9]+;\s*?$"), "", null, 0),
155 // struct TreeElement { }
156 // struct TreeElement { };
157 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)(\s+){([\sa-zA-Z0-9;:_]+?)}([~;])", "$1
→ $2$3{$4};$5", null, 0),
158 // class Program { }
159 // class Program { };
160 (new Regex(@"(struct|class) ([a-zA-Z0-9]+)[^\r\n]*([\r\n]+(?<indentLevel>\t
→ ]*)?)\{([\S\s]+?[\r\n]+\k<indentLevel>)\}([~;]|$)", "$1 $2$3{$4};$5", null, 0),
161 // class SizedBinaryTreeMethodsBase : GenericCollectionMethodsBase
162 // class SizedBinaryTreeMethodsBase : public GenericCollectionMethodsBase
163 (new Regex(@"class ([a-zA-Z0-9]+) : ([a-zA-Z0-9]+)", "class $1 : public $2", null,
→ 0),
164 // class IProperty : ISetter<TValue, TObject>, IProvider<TValue, TObject>
165 // class IProperty : public ISetter<TValue, TObject>, IProvider<TValue, TObject>
166 (new Regex(@"(?<before>class [a-zA-Z0-9]+ : ((public [a-zA-Z0-9]+(<[a-zA-Z0-9
→ ,]+)?, )+)?)(?<inheritedType>(?!(public) [a-zA-Z0-9]+(<[a-zA-Z0-9
→ ,]+)?)(?<after>([a-zA-Z0-9]+(?![>]| [\\r\\n]+)))", "${before}public
→ ${inheritedType}${after}", null, 10),
167 // Insert scope borders.
168 // ref TElement root
169 // ~!root!~ref TElement root
170 (new Regex(@"(?<definition>(?!<=|\\() (ref [a-zA-Z0-9]+|[a-zA-Z0-9]+(?<ref>))
→ (?<variable>[a-zA-Z0-9]+)(?=\\|,|=)"))", "~!${variable}!~${definition}", null,
→ 0),
171 // Inside the scope of ~!root!~ replace:
172 // root
173 // *root
174 (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),
175 // Remove scope borders.
176 // ~!root!~
177 //
178 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
179 // ref auto root = ref

```

```

180 // ref auto root =
181 (new Regex(@"ref ([a-zA-Z0-9]+) ([a-zA-Z0-9]+) = ref(\W)", "$1* $2 =$3", null, 0),
182 // *root = ref left;
183 // root = left;
184 (new Regex(@"\"*([a-zA-Z0-9]+) = ref ([a-zA-Z0-9]+)(\W)", "$1 = $2$3", null, 0),
185 // (ref left)
186 // (left)
187 (new Regex(@"\"(ref ([a-zA-Z0-9]+)(\\|\\(|,))", "($1$2", null, 0),
188 // ref TElement
189 // TElement*
190 (new Regex(@"\"(|\\()ref ([a-zA-Z0-9]+) \"", "$1$2* \"", null, 0),
191 // ref sizeBalancedTree.Root
192 // &sizeBalancedTree->Root
193 (new Regex(@"ref ([a-zA-Z0-9]+)\\.([a-zA-Z0-9\\*]+)", "&$1->$2", null, 0),
194 // ref GetElement(node).Right
195 // &GetElement(node)->Right
196 (new Regex(@"ref ([a-zA-Z0-9]+)\\((([a-zA-Z0-9\\*]+)\\)\\.([a-zA-Z0-9]+)\"",
197     ↳ "&$1($2)->$3", null, 0),
198 // GetElement(node).Right
199 // GetElement(node)->Right
200 (new Regex(@"([a-zA-Z0-9]+)\\((([a-zA-Z0-9\\*]+)\\)\\.([a-zA-Z0-9]+)\"", "$1($2)->$3",
201     ↳ null, 0),
202 // [Fact]public static void SizeBalancedTreeMultipleAttachAndDetachTest()
203 // TEST_METHOD(SizeBalancedTreeMultipleAttachAndDetachTest)
204 (new Regex(@"[Fact\\][\\s\\n]+(static)?void ([a-zA-Z0-9]+)\\(\\)", "TEST_METHOD($2)",
205     ↳ 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 // TElement Root;
213 // TElement Root = 0;
214 (new Regex(@"(\\r?\\n[\\t ]+)([a-zA-Z0-9:_]+(?<!return)) ([_a-zA-Z0-9]+);\"", "$1$2 $3 =
215     ↳ 0;", null, 0),
216 // TreeElement _elements[N];
217 // TreeElement _elements[N] = { {0} };
218 (new Regex(@"(\\r?\\n[\\t ]+)([a-zA-Z0-9]+) ([_a-zA-Z0-9]+)\\([([_a-zA-Z0-9]+)\\];\"",
219     ↳ "$1$2 $3[$4] = { {0} };", null, 0),
220 // auto path = new TElement[MaxPath];
221 // TElement path[MaxPath] = { {0} };
222 (new Regex(@"(\\r?\\n[\\t ]+)[a-zA-Z0-9]+ ([a-zA-Z0-9]+) = new
223     ↳ ([a-zA-Z0-9]+)\\([([_a-zA-Z0-9]+)\\];\"", "$1$3 $2[$4] = { {0} };", null, 0),
224 // Insert scope borders.
225 // auto added = new HashSet<TElement>();
226 // ~!added!~std::unordered_set<TElement> added;
227 (new Regex(@"auto (?<variable>[a-zA-Z0-9]+) = new
228     ↳ HashSet<(?<element>[a-zA-Z0-9]+)>\\(\\);\"",
229     ↳ "~!${variable}!~std::unordered_set<${element}> ${variable};", null, 0),
230 // Inside the scope of ~!added!~ replace:
231 // added.Add(node)
232 // added.insert(node)
233 (new Regex(@"\"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>\\.|\\n)(?<before>((?<
234     ↳ !~!\\k<variable>!~)(\\.|\\n)*?)\\k<variable>\\.Add\\((?<argument>[a-zA-Z0-9]+)\\)\")",
235     ↳ "${scope}${separator}${before}${variable}.insert(${argument})", null, 10),
236 // Inside the scope of ~!added!~ replace:
237 // added.Remove(node)
238 // added.erase(node)
239 (new Regex(@"\"(?<scope>~!(?<variable>[a-zA-Z0-9]+)!~)(?<separator>\\.|\\n)(?<before>((?<
240     ↳ !~!\\k<variable>!~)(\\.|\\n)*?)\\k<variable>\\.Remove\\((?<argument>[a-zA-Z0-9]+)\\)\")",
241     ↳ "${scope}${separator}${before}${variable}.erase(${argument})", null, 10),
242 // if (added.insert(node)) {
243 // if (!added.contains(node)) { added.insert(node);
244 (new Regex(@"\"if \\((?<variable>[a-zA-Z0-9]+)\\.insert\\((?<argument>[a-zA-Z0-9]+)\\)\\)\"(?
245     ↳ <separator>[\\t ]*[\\r\\n]+)(?<indent>[\\t ]*){\"", "if
246     ↳ (!${variable}.contains(${argument}))${separator}${indent}{\" +
247     ↳ Environment.NewLine + "${indent}    ${variable}.insert(${argument});", null, 0),
248 // Remove scope borders.
249 // ~!added!~
250 //
251 (new Regex(@"\"~!(?<pointer>[a-zA-Z0-9]+)!~\"", "", null, 5),
252 // Insert scope borders.
253 // auto random = new System.Random(0);
254 // std::srand(0);
255 (new Regex(@"\"[a-zA-Z0-9\\.]+ ([a-zA-Z0-9]+) = new
256     ↳ (System\\.\\.)?Random\\((([a-zA-Z0-9]+)\\);\"", "~!$1!~std::srand($3);", null, 0),

```

```

241 // Inside the scope of ~!random!~ replace:
242 // random.Next(1, N)
243 // (std::rand() % N) + 1
244 (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),
245 // Remove scope borders.
246 // ~!random!~
247 //
248 (new Regex(@"~!(?<pointer>[a-zA-Z0-9]+)!~"), "", null, 5),
249 // Insert method body scope starts.
250 // void PrintNodes(TElement node, StringBuilder sb, int level) {
251 // void PrintNodes(TElement node, StringBuilder sb, int level) { /*method-start*/
252 (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}${method}
    ↳ override}${arguments}${override}${separator}{ /*method-start*/${end}", null,
    ↳ 0),
253 // Insert method body scope ends.
254 // { /*method-start*/...}
255 // { /*method-start*/... /*method-end*/}
256 (new Regex(@"\{ /*method-start*/ (?<body>((?<bracket>\{) | (?<-bracket>\}) | [^\{\}]+) )
    ↳ \}"), "{ /*method-start*/ ${body} /*method-end*/", null,
    ↳ 0),
257 // Inside method bodies replace:
258 // GetFirst(
259 // this->GetFirst(
260 // (new Regex(@"(?<separator>(\(| |([W]) |return ))(?<!(-->|\*
    ↳ ))(?<method>(?!sizeof)[a-zA-Z0-9]+\)((?!\\) \{)"),
    ↳ "${separator}this->${method}(", null, 1),
261 (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),
262 // Remove scope borders.
263 // /*method-start*/
264 //
265 (new Regex(@"\/\*method-(start|end)\*\/"), "", null, 0),
266 // throw new ArgumentNullException(argumentName, message);
267 // throw std::invalid_argument(((std::string)"Argument
    ↳ ").append(argumentName).append(" is null: ").append(message).append("."));
268 (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),
269 // throw new ArgumentException(message, argumentName);
270 // throw std::invalid_argument(((std::string)"Invalid
    ↳ ").append(argumentName).append(" argument: ").append(message).append("."));
271 (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),
272 // throw new NotSupportedException();
273 // throw std::logic_error("Not supported exception.");
274 (new Regex(@"throw new NotSupportedException\(\);", "throw std::logic_error(\"Not
    ↳ supported exception.\");", null, 0),
275 // throw new NotImplementedException();
276 // throw std::logic_error("Not implemented exception.");
277 (new Regex(@"throw new NotImplementedException\(\);", "throw std::logic_error(\"Not
    ↳ implemented exception.\");", null, 0),
278
279 }.Cast<ISubstitutionRule>().ToList();
280
281 public static readonly IList<ISubstitutionRule> LastStage = new List<SubstitutionRule>
282 {
283     // ICounter<int, int> c1;
284     // ICounter<int, int>* c1;
285     (new Regex(@"(?<abstractType>I[A-Z][a-zA-Z0-9]+(<[~>\r\n]+>)?
    ↳ (?<variable>[_a-zA-Z0-9]+);", "${abstractType}* ${variable};", null, 0),
286     // (expression)
287     // expression
288     (new Regex(@"(\(| \)|)(([a-zA-Z0-9_]*:)+)\(| |;|\\)"), "$1$2$3", null, 0),
289     // (method(expression))
290     // method(expression)

```

```

291         (new Regex(@"(?<firstSeparator>\(|
    ↪    )\|(?<method>[a-zA-Z0-9_\->\*:]+)\|((?<expression>((?<parenthesis>\(|(?<-parent
    ↪    hesis>))|[a-zA-Z0-9_\->\*:]*)+)(?(parenthesis)(?!))\|)\|(?<lastSeparator>(|
    ↪    |;|\\))"), "${firstSeparator}${method}(${expression})${lastSeparator}", null, 0),
292     // return ref _elements[node];
293     // return &_elements[node];
294     (new Regex(@"return ref ([_a-zA-Z0-9]+)\|([_a-zA-Z0-9\*]+)\|;"), "return &$1[$2];",
    ↪     null, 0),
295     // default
296     // 0
297     (new Regex(@"(\W)default(\W)"), "${1}0$2", null, 0),
298     // // #define ENABLE_TREE_AUTO_DEBUG_AND_VALIDATION
299     //
300     (new Regex(@"\\\/[ \t]*#define[ \t]+[_a-zA-Z0-9]+[ \t]*"), "", null, 0),
301     // #if USEARRAYPOOL\r\n#endif
302     //
303     (new Regex(@"#if [a-zA-Z0-9]+\s+#endif"), "", null, 0),
304     // [Fact]
305     //
306     (new Regex(@"(?<firstNewLine>\r?\n|\A)(?<indent>[ \t
    ↪    ]+)\|([_a-zA-Z0-9]+(\|((?<expression>((?<parenthesis>\(|(?<-parent
    ↪    hesis>))|[a-zA-Z0-9_\->\*:]*)+)(?(parenthesis)(?!))\|)\|(?<lastSeparator>(|
    ↪    |;|\\))\|)\|([ \t]*\r?\n\k<indent>)?"),
    ↪     "${firstNewLine}${indent}", null, 5),
307     // \n ... namespace
308     // namespace
309     (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+namespace"), "$1namespace", null, 0),
310     // \n ... class
311     // class
312     (new Regex(@"(\\S[\\r\\n]{1,2})?[\\r\\n]+class"), "$1class", null, 0),
313     }.Cast<ISubstitutionRule>().ToList();
314
315     public CSharpToCppTransformer(IList<ISubstitutionRule> extraRules) :
    ↪     base(FirstStage.Concat(extraRules).Concat(LastStage).ToList()) { }
316
317     public CSharpToCppTransformer() : base(FirstStage.Concat(LastStage).ToList()) { }
318 }
319 }

```

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