

Understand Metrics

Table of Contents

1. [All Methods](#)
2. [Auto-Implemented Properties](#)
3. [Average Blank Lines](#)
4. [Average Blank Lines \(Includes Inactive\)](#)
5. [Average Code Lines](#)
6. [Average Code Lines \(Includes Inactive\)](#)
7. [Average Comment Lines](#)
8. [Average Comment Lines \(Includes Inactive\)](#)
9. [Average Cyclomatic Complexity](#)
10. [Average Essential Complexity](#)
11. [Average Lines](#)
12. [Average Modified Cyclomatic Complexity](#)
13. [Average Strict Cyclomatic Complexity](#)
14. [Average Strict Modified Cyclomatic Complexity](#)
15. [Average Strict Modified Essential Complexity](#)
16. [Base Classes](#)
17. [Blank Lines](#)
18. [Blank Lines \(HTML\)](#)
19. [Blank Lines \(Includes Inactive\)](#)
20. [Blank Lines \(JavaScript\)](#)
21. [Blank Lines \(PHP\)](#)
22. [Changed Lines](#)
23. [Class Methods](#)
24. [Class Variables](#)
25. [Classes](#)
26. [Code Files](#)
27. [Code Lines](#)
28. [Code Lines \(Includes Inactive\)](#)
29. [Code Lines \(JavaScript\)](#)
30. [Code Lines \(PHP\)](#)
31. [CodeCheck Violation Density by Code Lines](#)
32. [CodeCheck Violation Density by Lines](#)
33. [CodeCheck Violation Types](#)
34. [CodeCheck Violations](#)
35. [Comment Lines](#)
36. [Comment Lines \(HTML\)](#)
37. [Comment Lines \(Includes Inactive\)](#)
38. [Comment Lines \(JavaScript\)](#)
39. [Comment Lines \(PHP\)](#)
40. [Comment to Code Ratio](#)

41. [Const Methods](#)
42. [Coupled Classes](#)
43. [Coupled Classes Modified](#)
44. [Coupled Packages](#)
45. [Cyclomatic Complexity](#)
46. [Declarative Code Lines](#)
47. [Declarative Statements](#)
48. [Declarative Statements \(Javascript\)](#)
49. [Declarative Statements \(PHP\)](#)
50. [Default Methods](#)
51. [Derived Classes](#)
52. [Empty Statements](#)
53. [Essential Complexity](#)
54. [Executable Code Lines](#)
55. [Executable Statements](#)
56. [Executable Statements \(JavaScript\)](#)
57. [Executable Statements \(PHP\)](#)
58. [Executable Units](#)
59. [Files](#)
60. [Friend Methods](#)
61. [Functions](#)
62. [Header Files](#)
63. [Inactive Lines](#)
64. [Inputs](#)
65. [Instance Methods](#)
66. [Instance Variables](#)
67. [Internal Instance Variables](#)
68. [Internal Methods](#)
69. [Knots](#)
70. [Lines](#)
71. [Lines \(HTML\)](#)
72. [Lines \(JavaScript\)](#)
73. [Lines \(PHP\)](#)
74. [Max Cyclomatic Complexity](#)
75. [Max Essential Complexity](#)
76. [Max Essential Knots](#)
77. [Max Inheritance Tree](#)
78. [Max Modified Cyclomatic Complexity](#)
79. [Max Nesting](#)
80. [Max Strict Cyclomatic Complexity](#)
81. [Max Strict Modified Cyclomatic Complexity](#)
82. [Max Strict Modified Essential Complexity](#)
83. [Methods](#)
84. [Min Essential Knots](#)
85. [Modified Cyclomatic Complexity](#)

- 86. Modules
- 87. New Lines
- 88. Outputs
- 89. Paths
- 90. Paths Log(x)
- 91. Percent Changed
- 92. Percent Lack Of Cohesion
- 93. Percent Lack Of Cohesion Modified
- 94. Preprocessor Lines
- 95. Private Instance Variables
- 96. Private Methods
- 97. Program Units
- 98. Properties
- 99. Protected Instance Variables
- 100. Protected Internal Instance Variables
- 101. Protected Internal Methods
- 102. Protected Methods
- 103. Public Instance Variables
- 104. Public Methods
- 105. Removed Lines
- 106. Semicolons
- 107. Statements
- 108. Strict Cyclomatic Complexity
- 109. Strict Modified Cyclomatic Complexity
- 110. Strict Modified Essential Complexity
- 111. Strict Private Methods
- 112. Strict Published Methods
- 113. Subprograms
- 114. Sum Cyclomatic Complexity
- 115. Sum Essential Complexity
- 116. Sum Modified Cyclomatic Complexity
- 117. Sum Strict Cyclomatic Complexity
- 118. Sum Strict Modified Cyclomatic Complexity
- 119. Sum Strict Modified Essential Complexity

All Methods

API ID: CountDeclMethodAll

Also Known As: Number of Methods (NM), Response for a Class (RFC)

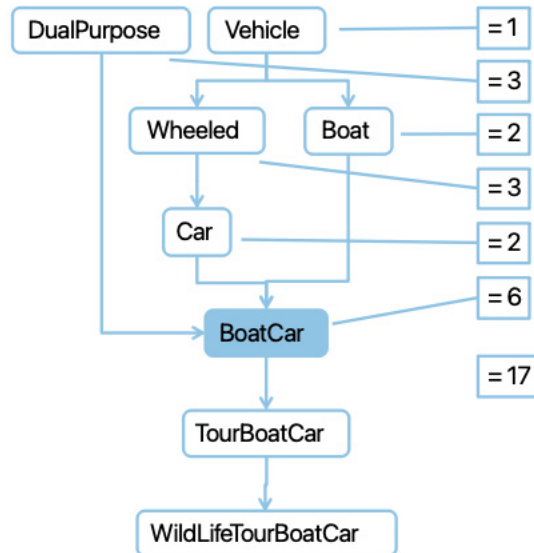
Languages: Basic, C#, C++, Java, Pascal, Python, Web

Targets: Classes

Number of methods, including inherited ones.

Also known as Chidamber & Kemerer - Response for a Class (RFC); Lorenz & Kidd -

Number of Methods (NM).



Targets By Language:

- **Basic:** Class, Struct
- **C#:** Class, Struct
- **C++:** Class, Struct, Union
- **Java:** Class, Interface
- **Pascal:** Class, Interface
- **Python:** Class
- **Web:** PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Auto-Implemented Properties

API ID: CountDeclPropertyAuto

Languages: C#

Targets: Architectures, Classes, Project

Number of auto-implemented properties.

Targets By Language:

- **C#:** Project, Class, Struct

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

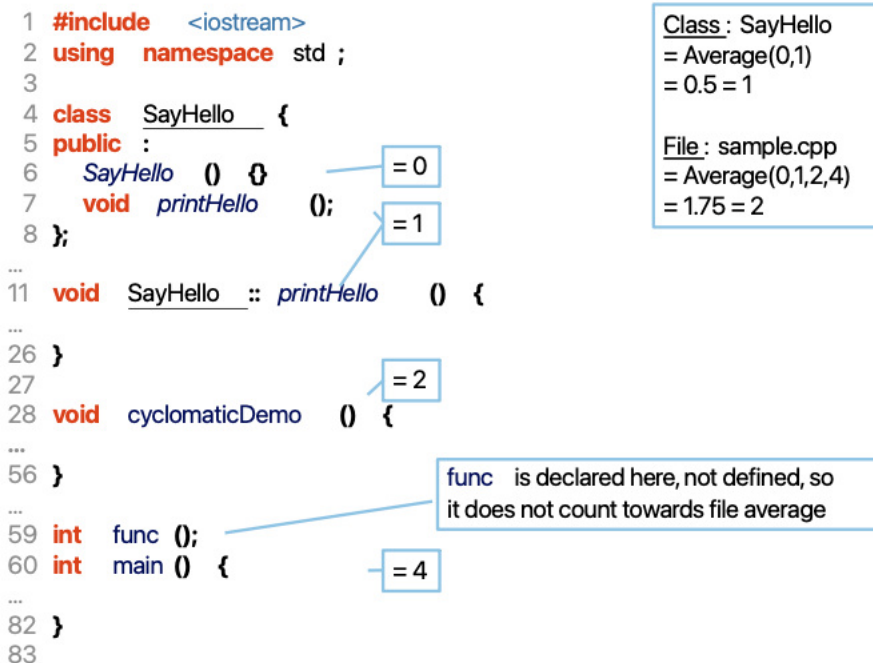
Average Blank Lines

API ID: AvgCountLineBlank

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Classes, Files, Modules, Packages

Average number of blank lines for all nested functions or methods.



```

1  #include <iostream>
2  using namespace std ;
3
4  class SayHello {
5  public :
6      SayHello () {}
7      void printHello () ;
8  };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {
...
56 }
...
59 int func ();
60 int main () {
...
82 }
83

```

Class: SayHello
 = Average(0,1)
 = 0.5 = 1

File: sample.cpp
 = Average(0,1,2,4)
 = 1.75 = 2

func is declared here, not defined, so it does not count towards file average

Targets By Language:

- **Ada:** File, Package
- **Basic:** File, Module, Class, Struct
- **C#:** File, Class, Struct
- **C++:** File, Class, Struct, Union
- **Fortran:** File

- **Java:** File, Class, Interface
- **Jovial:** File
- **Pascal:** File, Class, Interface
- **Python:** File, Class
- **Rust:** File
- **Web:** File, PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Include inactive lines" option for languages that have inactive lines.

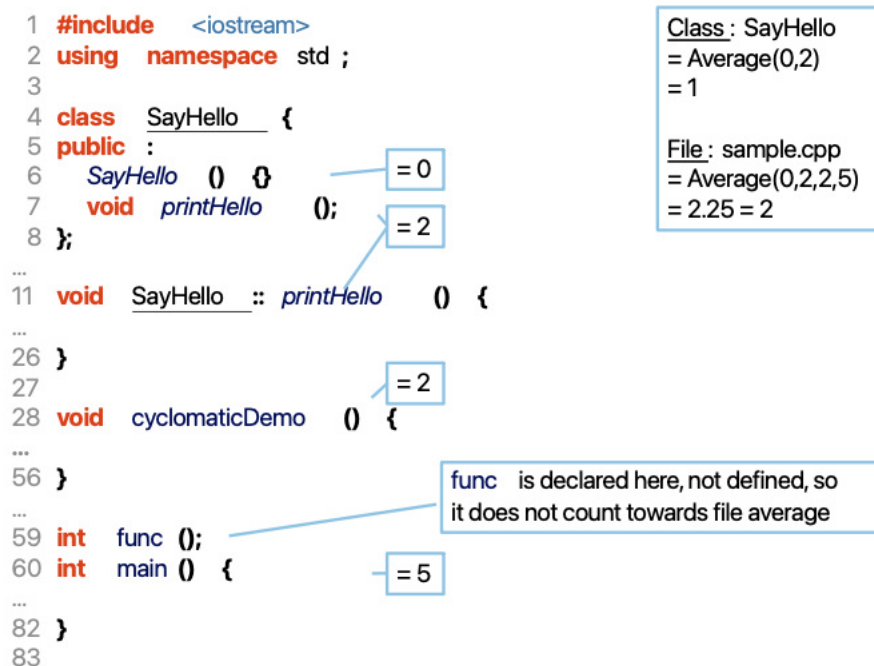
Average Blank Lines (Includes Inactive)

API ID: AvgCountLineBlankWithInactive

Languages: C++

Targets: Classes, Files

Average number of blank lines for all nested functions or methods, including inactive regions.



```

1  #include <iostream>
2  using namespace std ;
3
4  class SayHello {
5  public :
6      SayHello () {}           = 0
7      void printHello () ;     = 2
8  };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {     = 2
...
56 }
...
59 int func ();
60 int main () {                = 5
...
82 }
83

```

Class : SayHello
 = Average(0,2)
 = 1

File : sample.cpp
 = Average(0,2,2,5)
 = 2.25 = 2

func is declared here, not defined, so it does not count towards file average

Targets By Language:

- **C++:** File, Class, Struct, Union

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Include inactive lines" option.

Average Code Lines

API ID: AvgCountLineCode

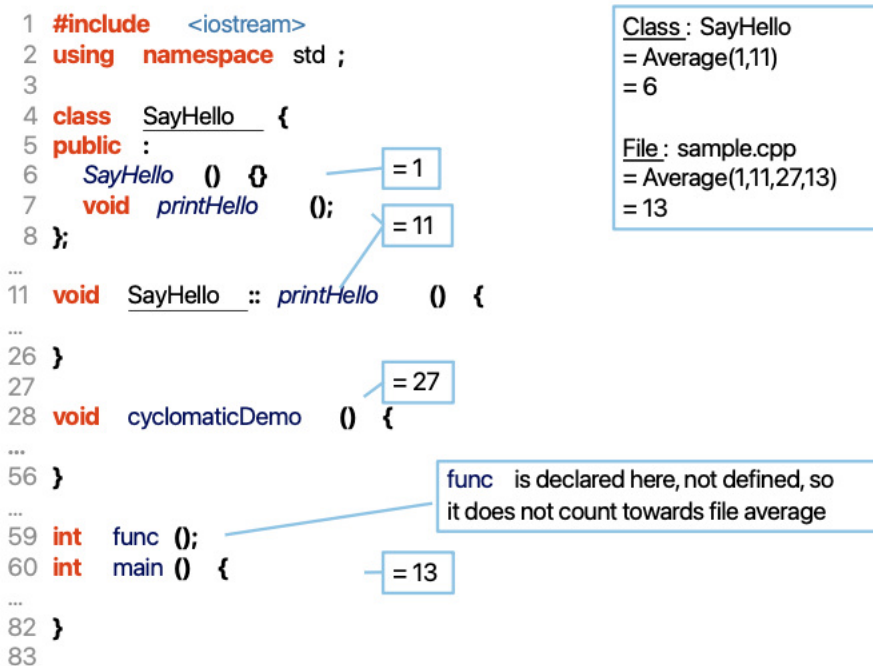
Also Known As: Average Method Size (AMS)

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Classes, Files, Modules, Packages

Average number of lines containing source code for all nested functions or methods.

Also known as Lorenz & Kidd - Average Method Size (AMS).



```

1  #include <iostream>
2  using namespace std ;
3
4  class SayHello {
5  public :
6      SayHello () {}           = 1
7      void printHello () {}    = 11
8  };
...
11 void SayHello :: printHello () {
...
26 }
27                               = 27
28 void cyclomaticDemo () {
...
56 }
...
59 int func ();
60 int main () {                = 13
...
82 }
83

```

Class : SayHello
= Average(1,11)
= 6

File : sample.cpp
= Average(1,11,27,13)
= 13

func is declared here, not defined, so it does not count towards file average

Targets By Language:

- **Ada:** File, Package
- **Basic:** File, Module, Class, Struct
- **C#:** File, Class, Struct
- **C++:** File, Class, Struct, Union

- **Fortran:** File
- **Java:** File, Class, Interface
- **Jovial:** File
- **Pascal:** File, Class, Interface
- **Python:** File, Class
- **Rust:** File
- **Web:** File, PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Include inactive lines" option for languages that have inactive lines.

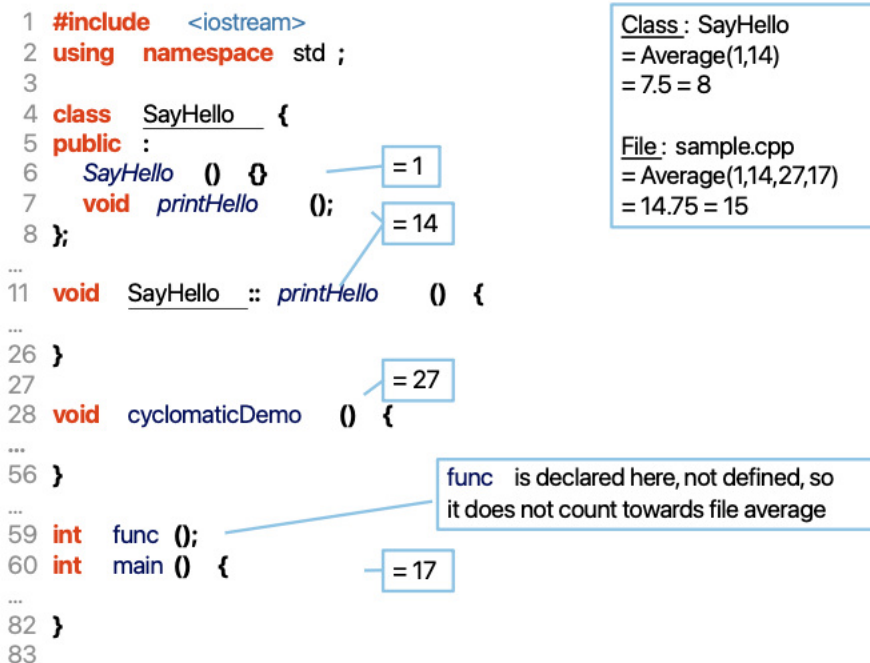
Average Code Lines (Includes Inactive)

API ID: AvgCountLineCodeWithInactive

Languages: C++

Targets: Classes, Files

Average number of lines containing source code for all nested functions or methods, including inactive regions.



```

1 #include <iostream>
2 using namespace std ;
3
4 class SayHello {
5 public :
6     SayHello () {}
7     void printHello () ;
8 };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {
...
56 }
...
59 int func () ;
60 int main () {
...
82 }
83

```

Annotations:

- Line 6: `SayHello () {}` → = 1
- Line 7: `void printHello () ;` → = 14
- Line 28: `void cyclomaticDemo () {` → = 27
- Line 60: `int main () {` → = 17

Summary:

- Class : SayHello
= Average(1,14)
= 7.5 = 8
- File : sample.cpp
= Average(1,14,27,17)
= 14.75 = 15

Note: `func` is declared here, not defined, so it does not count towards file average

Targets By Language:

- **C++:** File, Class, Struct, Union

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Include inactive lines" option.

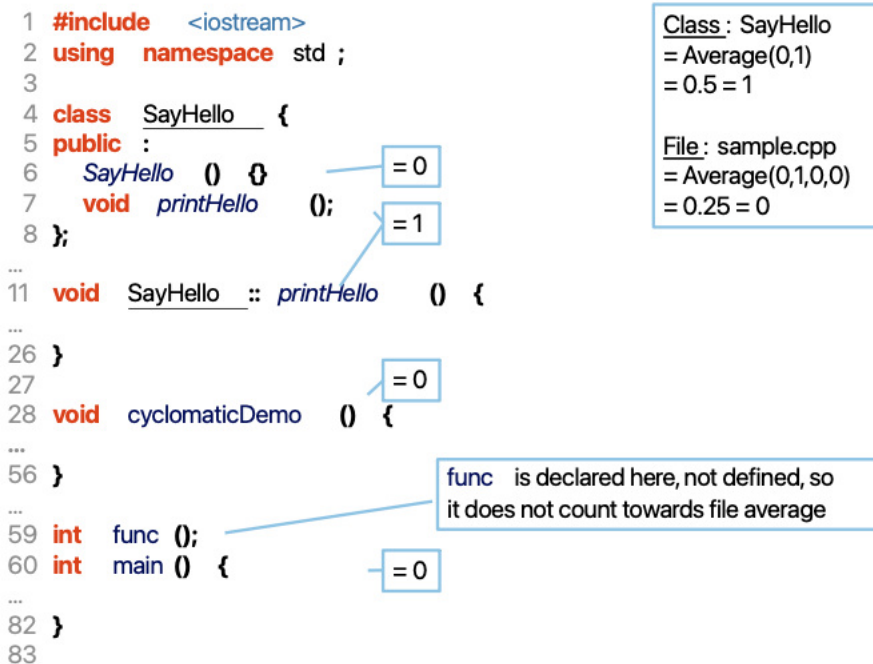
Average Comment Lines

API ID: AvgCountLineComment

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Classes, Files, Modules, Packages

Average number of lines containing comment for all nested functions or methods.



```

1  #include <iostream>
2  using namespace std ;
3
4  class SayHello {
5  public :
6      SayHello () {}
7      void printHello () ;
8  };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {
...
56 }
...
59 int func ();
60 int main () {
...
82 }
83

```

Class : SayHello
 = Average(0,1)
 = 0.5 = 1

File : sample.cpp
 = Average(0,1,0,0)
 = 0.25 = 0

func is declared here, not defined, so it does not count towards file average

Targets By Language:

- **Ada:** File, Package
- **Basic:** File, Module, Class, Struct
- **C#:** File, Class, Struct
- **C++:** File, Class, Struct, Union
- **Fortran:** File

- **Java:** File, Class, Interface
- **Jovial:** File
- **Pascal:** File, Class, Interface
- **Python:** File, Class
- **Rust:** File
- **Web:** File, PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Include inactive lines" option for languages that have inactive lines.

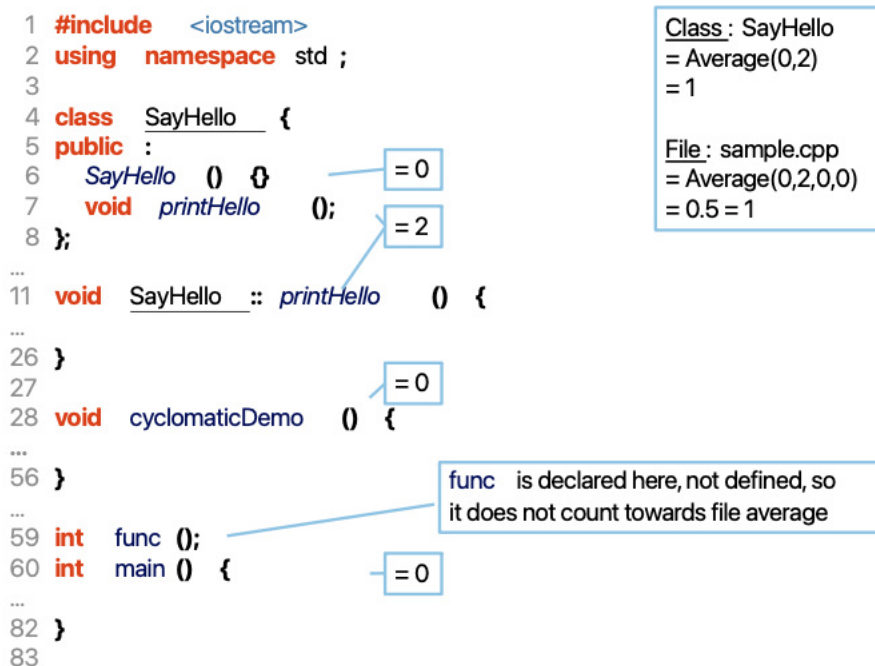
Average Comment Lines (Includes Inactive)

API ID: AvgCountLineCommentWithInactive

Languages: C++

Targets: Classes, Files

Average number of lines containing comment for all nested functions or methods, including inactive regions.



```

1  #include <iostream>
2  using namespace std ;
3
4  class SayHello {
5  public :
6      SayHello () {}
7      void printHello () ;
8  };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {
...
56 }
...
59 int func ();
60 int main () {
...
82 }
83

```

Class : SayHello
 = Average(0,2)
 = 1

File : sample.cpp
 = Average(0,2,0,0)
 = 0.5 = 1

func is declared here, not defined, so it does not count towards file average

Targets By Language:

- **C++:** File, Class, Struct, Union

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Include inactive lines" option.

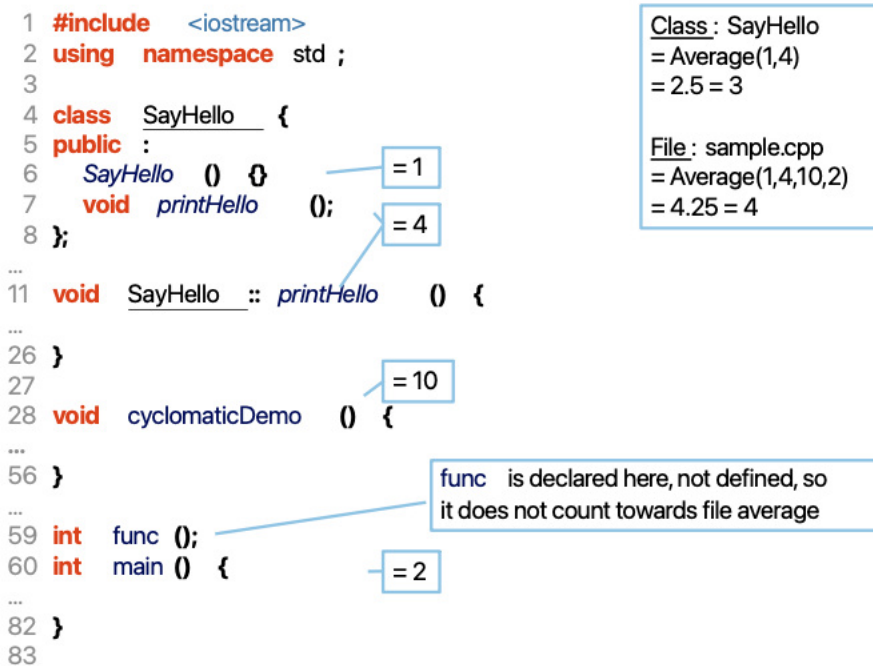
Average Cyclomatic Complexity

API ID: AvgCyclomatic

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, VHDL, Web

Targets: Architectures, Classes, Files, Modules, Packages, Project

Average cyclomatic complexity for all nested functions or methods.



```

1 #include <iostream>
2 using namespace std ;
3
4 class SayHello {
5 public :
6     SayHello () {}
7     void printHello () ;
8 };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {
...
56 }
...
59 int func () ;
60 int main () {
...
82 }
83

```

Class : SayHello
= Average(1,4)
= 2.5 = 3

File : sample.cpp
= Average(1,4,10,2)
= 4.25 = 4

func is declared here, not defined, so it does not count towards file average

Targets By Language:

- **Ada:** Project, File, Package
- **Basic:** Project, File, Module, Class, Struct
- **C#:** Project, File, Class, Struct
- **C++:** Project, File, Class, Struct, Union
- **Fortran:** Project, File

- **Java:** Project, File, Class, Interface
- **Jovial:** Project, File
- **Pascal:** Project, File, Class, Interface, Compunit
- **Python:** Project, File, Class
- **Rust:** Project, File
- **VHDL:** Project, File, Architecture
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- The cyclomatic metric used is configured from "Project Configuration/Metrics/Complexity". This metric results from having "Count each decision in a multi-way-decision structure" on and having "Count logical conjunctions (OR) and logical and (AND)" off. Checking "Show all available cyclomatic complexity metric variants" will also enable the metric.

Average Essential Complexity

API ID: AvgEssential

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Python, Rust, Web

Targets: Architectures, Classes, Files, Modules, Packages, Project

Average Essential complexity for all nested functions or methods.

```

1  #include <iostream>
2  using namespace std ;
3
4  class SayHello {
5  public :
6      SayHello () {}
7      void printHello () ;
8  };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {
...
56 }
...
59 int func () ;
60 int main () {
...
82 }
83

```

Annotations for Cyclomatic Complexity:

- Line 6: `SayHello () {}` → = 1
- Line 7: `void printHello () ;` → = 3
- Line 28: `void cyclomaticDemo () {` → = 1
- Line 59: `int func () ;` → = 1

Class : SayHello
= Average(1,3)
= 2

File : sample.cpp
= Average(1,3,1,1)
= 1.5

func is declared here, not defined, so it does not count towards file average

Targets By Language:

- **Ada:** Project, File, Package
- **Basic:** Project, File, Module, Class, Struct
- **C#:** Project, File, Class, Struct
- **C++:** Project, File, Class, Struct, Union
- **Fortran:** Project, File
- **Java:** Project, File, Class, Interface
- **Jovial:** Project, File
- **Python:** Project, File, Class
- **Rust:** Project, File
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Complexity" with the "Essential" option.

Average Lines

API ID: AvgCountLine

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Classes, Files, Modules, Packages

Average number of lines for all nested functions or methods.

```

1  #include <iostream>
2  using namespace std ;
3
4  class SayHello {
5  public :
6      SayHello () {}           = 1
7      void printHello () ;     = 16
8  };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {     = 29
...
56 }
...
59 int func ();
60 int main () {                = 23
...
82 }
83

```

Class : SayHello
 = Average(1,16)
 = 8.5 = 9

File : sample.cpp
 = Average(1,16,29,23)
 = 17.3 = 17

func is declared here, not defined, so
 it does not count towards file average

Targets By Language:

- **Ada:** File, Package
- **Basic:** File, Module, Class, Struct
- **C#:** File, Class, Struct
- **C++:** File, Class, Struct, Union
- **Fortran:** File
- **Java:** File, Class, Interface
- **Jovial:** File
- **Pascal:** File, Class, Interface
- **Python:** File, Class
- **Rust:** File
- **Web:** File, PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.

Average Modified Cyclomatic Complexity

API ID: AvgCyclomaticModified

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, VHDL, Web

Targets: Architectures, Classes, Files, Modules, Packages, Project

Average modified cyclomatic complexity for all nested functions or methods.

```

1  #include <iostream>
2  using namespace std ;
3
4  class SayHello {
5  public :
6      SayHello () {}
7      void printHello () ;
8  };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {
...
56 }
...
59 int func () ;
60 int main () {
...
82 }
83

```

Annotations for Cyclomatic Complexity:

- Line 6: `SayHello () {}` → = 1
- Line 7: `void printHello () ;` → = 3
- Line 28: `void cyclomaticDemo () {` → = 8
- Line 59: `int func () ;` → = 2

Class : SayHello
 = Average(1,3)
 = 2

File : sample.cpp
 = Average(1,3,8,2)
 = 3.5 = 4

func is declared here, not defined, so it does not count towards file average

Targets By Language:

- **Ada:** Project, File, Package
- **Basic:** Project, File, Module, Class, Struct
- **C#:** Project, File, Class, Struct
- **C++:** Project, File, Class, Struct, Union
- **Fortran:** Project, File
- **Java:** Project, File, Class, Interface
- **Jovial:** Project, File
- **Pascal:** Project, File, Class, Interface, Compunit
- **Python:** Project, File, Class
- **Rust:** Project, File
- **VHDL:** Project, File, Architecture
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- The cyclomatic metric used is configured from "Project Configuration/Metrics/Complexity". This metric results from having "Count each decision in

a multi-way-decision structure" off and having "Count logical conjunctions (OR) and logical and (AND)" off. Checking "Show all available cyclomatic complexity metric variants" will also enable the metric.

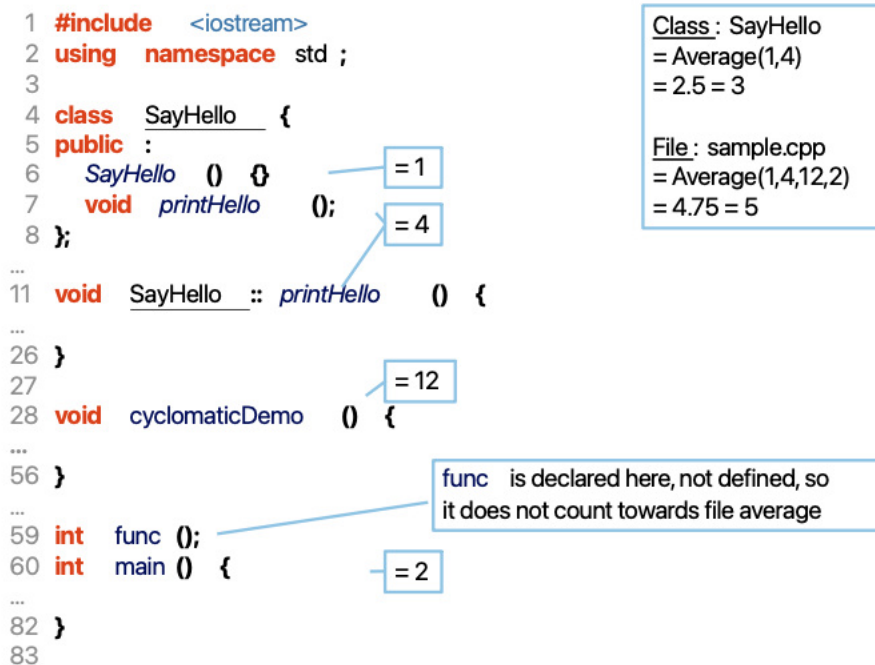
Average Strict Cyclomatic Complexity

API ID: AvgCyclomaticStrict

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Architectures, Classes, Files, Modules, Packages, Project

Average strict cyclomatic complexity for all nested functions or methods.



```

1 #include <iostream>
2 using namespace std ;
3
4 class SayHello {
5 public :
6     SayHello () {}
7     void printHello () {
8 };
9
10 ...
11 void SayHello :: printHello () {
12 ...
13 }
14
15 ...
16 void SayHello :: printHello () {
17 ...
18 }
19
20 ...
21 void SayHello :: printHello () {
22 ...
23 }
24
25 ...
26 }
27
28 void SayHello :: printHello () {
29 ...
30 }
31
32 ...
33 }
34
35 ...
36 }
37
38 ...
39 }
40
41 ...
42 }
43
44 ...
45 }
46
47 ...
48 }
49
50 ...
51 }
52
53 ...
54 }
55
56 ...
57 }
58
59 int func ();
60 int main () {
61 ...
62 }
63
64 ...
65 }
66
67 ...
68 }
69
70 ...
71 }
72
73 ...
74 }
75
76 ...
77 }
78
79 ...
80 }
81
82 }
83

```

Class : SayHello
= Average(1,4)
= 2.5 = 3

File : sample.cpp
= Average(1,4,12,2)
= 4.75 = 5

func is declared here, not defined, so it does not count towards file average

Targets By Language:

- **Ada:** Project, File, Package
- **Basic:** Project, File, Module, Class, Struct
- **C#:** Project, File, Class, Struct
- **C++:** Project, File, Class, Struct, Union
- **Fortran:** Project, File
- **Java:** Project, File, Class, Interface
- **Jovial:** Project, File
- **Pascal:** Project, File, Class, Interface, Compunit
- **Python:** Project, File, Class
- **Rust:** Project, File
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- The cyclomatic metric used is configured from "Project Configuration/Metrics/Complexity". This metric results from having "Count each decision in a multi-way-decision structure" on and having "Count logical conjunctions (OR) and logical and (AND)" on. Checking "Show all available cyclomatic complexity metric variants" will also enable the metric.

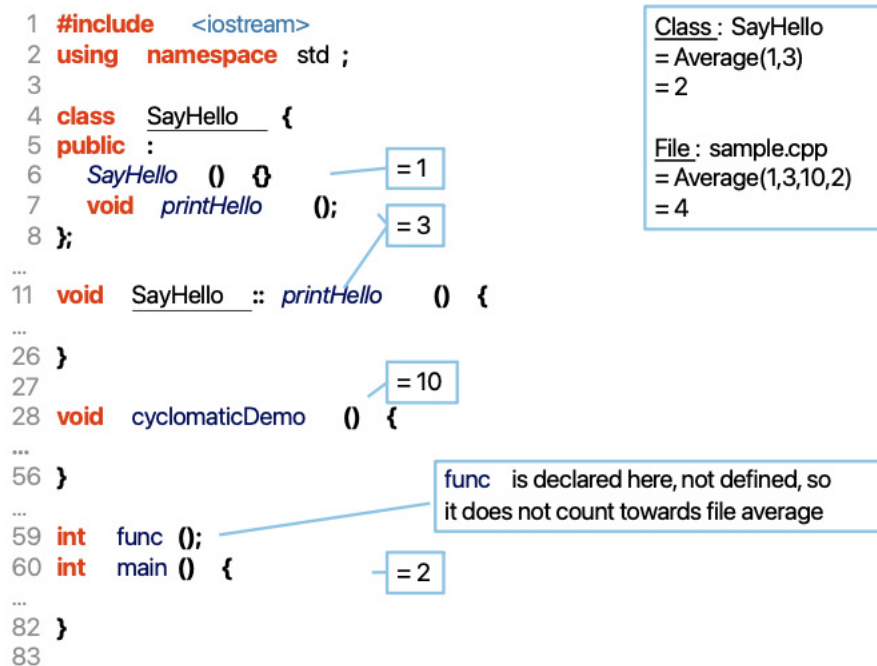
Average Strict Modified Cyclomatic Complexity

API ID: AvgCyclomaticStrictModified

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Architectures, Classes, Files, Modules, Packages, Project

Average strict modified cyclomatic complexity for all nested functions or methods.



```

1 #include <iostream>
2 using namespace std ;
3
4 class SayHello {
5 public :
6     SayHello () {}
7     void printHello () ;
8 };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {
...
56 }
...
59 int func () ;
60 int main () {
...
82 }
83

```

Class : SayHello
= Average(1,3)
= 2

File : sample.cpp
= Average(1,3,10,2)
= 4

func is declared here, not defined, so it does not count towards file average

Targets By Language:

- **Ada:** Project, File, Package
- **Basic:** Project, File, Module, Class, Struct
- **C#:** Project, File, Class, Struct
- **C++:** Project, File, Class, Struct, Union

- **Fortran:** Project, File
- **Java:** Project, File, Class, Interface
- **Jovial:** Project, File
- **Pascal:** Project, File, Class, Interface, Compunit
- **Python:** Project, File, Class
- **Rust:** Project, File
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- The cyclomatic metric used is configured from "Project Configuration/Metrics/Complexity". This metric results from having "Count each decision in a multi-way-decision structure" off and having "Count logical conjunctions (OR) and logical and (AND)" on. Checking "Show all available cyclomatic complexity metric variants" will also enable the metric.

Average Strict Modified Essential Complexity

API ID: AvgEssentialStrictModified

Languages: Ada

Targets: Architectures, Files, Packages, Project

Average strict modified essential complexity for all nested functions or methods.

Targets By Language:

- **Ada:** Project, File, Package

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Complexity" with the "Essential" option.

Base Classes

API ID: CountClassBase

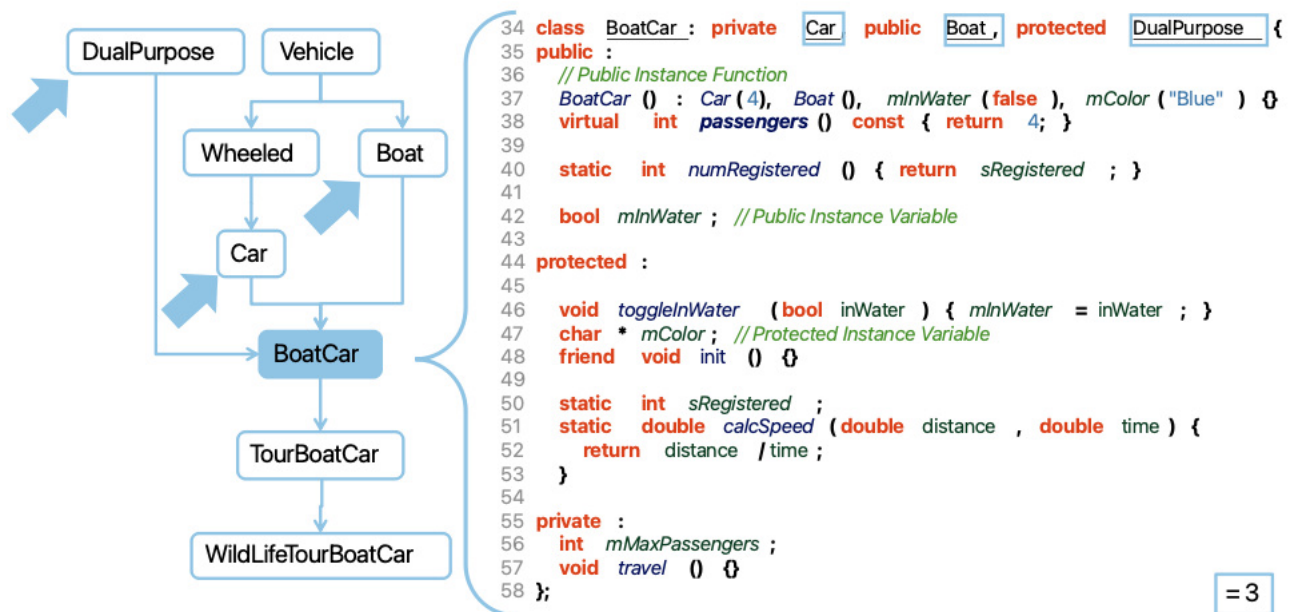
Also Known As: IFANIN

Languages: Basic, C#, C++, Java, Pascal, Python, Web

Targets: Classes

Number of immediate base classes.

Also known as IFANIN.



Targets By Language:

- **Basic:** Class, Struct
- **C#:** Class, Struct
- **C++:** Class, Struct, Union
- **Java:** Class, Interface
- **Pascal:** Class, Interface
- **Python:** Class
- **Web:** PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show inheritance metrics" option.

Blank Lines

API ID: CountLineBlank

Also Known As: Blank Lines of Code (BLOC)

Languages: Ada, Assembly, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, VHDL, Web

Also known as Blank Lines of Code (BLOC).

[illegible]

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Project, File, Method, Module, Class
- **C#:** Project, File, Class, Method
- **C++:** Project, File, Class, Struct, Union, Function
- **Fortran:** Project, File, Module, Block Data, Function, Program, Subroutine
- **Java:** Project, File, Class, Interface, Method
- **Jovial:** Project, File, Module, Subroutine
- **Pascal:** Project, File, Class, Interface, Compunit, Function, Procedure
- **Python:** Project, File, Class, Function
- **Rust:** Project, File, Function
- **VHDL:** Project, File, Procedure, Function, Process, Architecture
- **Web:** Project, File, PHP Class, PHP Interface

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Include inactive lines" option for languages that have inactive lines.
- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show default summary metrics" option.

Blank Lines (HTML)

API ID: CountLineBlankHtml

Languages: Web

Targets: Architectures, Files, Project

Number of blank HTML lines.

Targets By Language:

- **Web:** Project, File

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Breakdown counts by web language" option.

Blank Lines (Includes Inactive)

API ID: CountLineBlankWithInactive

Languages: C++

Targets: Architectures, Classes, Files, Functions, Project

Number of blank lines, including inactive regions.

Code	Comment	Preprocessor	Declarative	Executable	Inactive	
✓			✓			11 void SayHello :: <i>printHello</i> () {
✓				✓		12 switch (i) {
✓				✓		13 case 0:
✓				✓		14 cout << "Hello World" << endl ;
✓				✓		15 case 1:
✓				✓		16 cout << "HELLO WORLD!" << endl ;
✓	✓			✓		17 default : <i>// a comment here</i>
✓			✓	✓		18 for (int m = 0; m < j; m++);
✓				✓		19 cout << "hello world" << endl ;
✓						20 }
		✓				21 #ifdef A_VERY_NICE_VARIABLE
				✓		22 cout << "Inactive Line" << endl ; <i>// Inactive</i>
✓	✓			✓		23
				✓		24 #endif
						25
✓						26 }

= not (Code || Comment || Preprocessor)
= 2

Targets By Language:

- **C++:** Project, File, Class, Struct, Union, Function

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Include inactive lines" option.

Blank Lines (JavaScript)

API ID: CountLineBlankJavascript

Languages: Web

Targets: Architectures, Files, Project

Number of blank JavaScript lines.

Targets By Language:

- **Web:** Project, File

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.

- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Breakdown counts by web language" option.

Blank Lines (PHP)

API ID: CountLineBlankPhp

Languages: Web

Targets: Architectures, Classes, Files, Project

Number of blank PHP lines.

Targets By Language:

- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Breakdown counts by web language" option.

Changed Lines

API ID: CountLineChanged

Languages: Ada, Assembly, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, VHDL, Web

Targets: Classes, Files, Functions, Modules, Packages, Subprograms

Number of changed lines

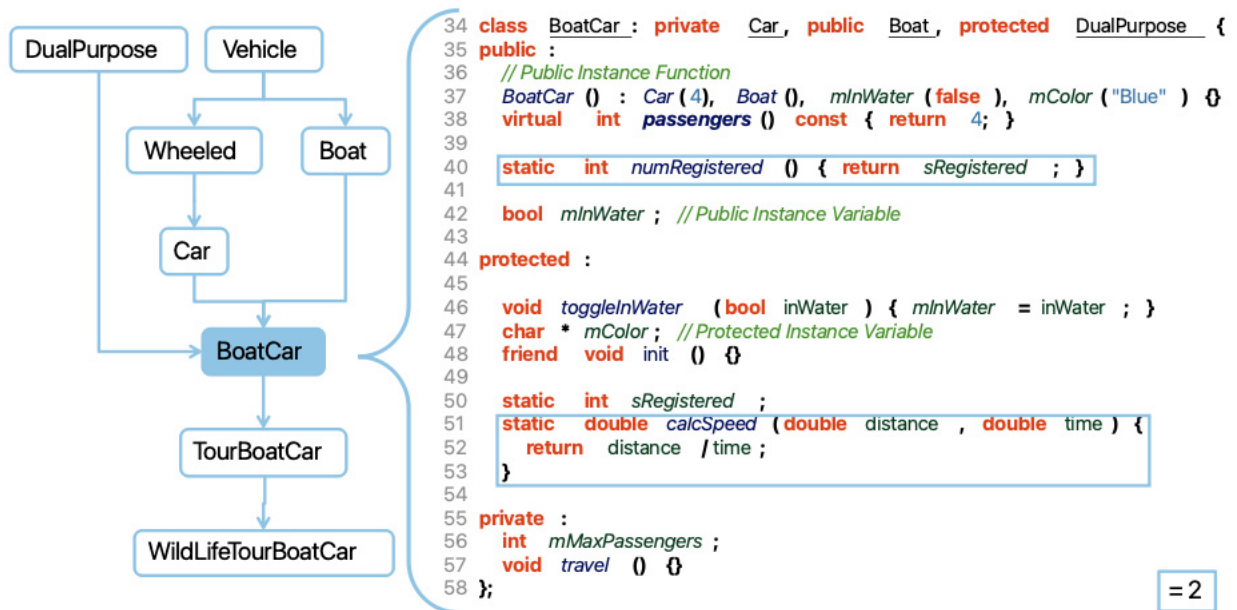
Class Methods

API ID: CountDeclClassMethod

Languages: Basic, C#, C++, Java, Pascal, Web

Targets: Architectures, Classes, Files, Project

Number of class methods.



= 2

Targets By Language:

- **Basic:** Project, Class, Struct
- **C#:** Project, Class, Struct
- **C++:** Project, Class, Struct, Union
- **Java:** Project, File, Class, Interface
- **Pascal:** Project, Class, Interface
- **Web:** Project, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Class Variables

API ID: CountDeclClassVariable

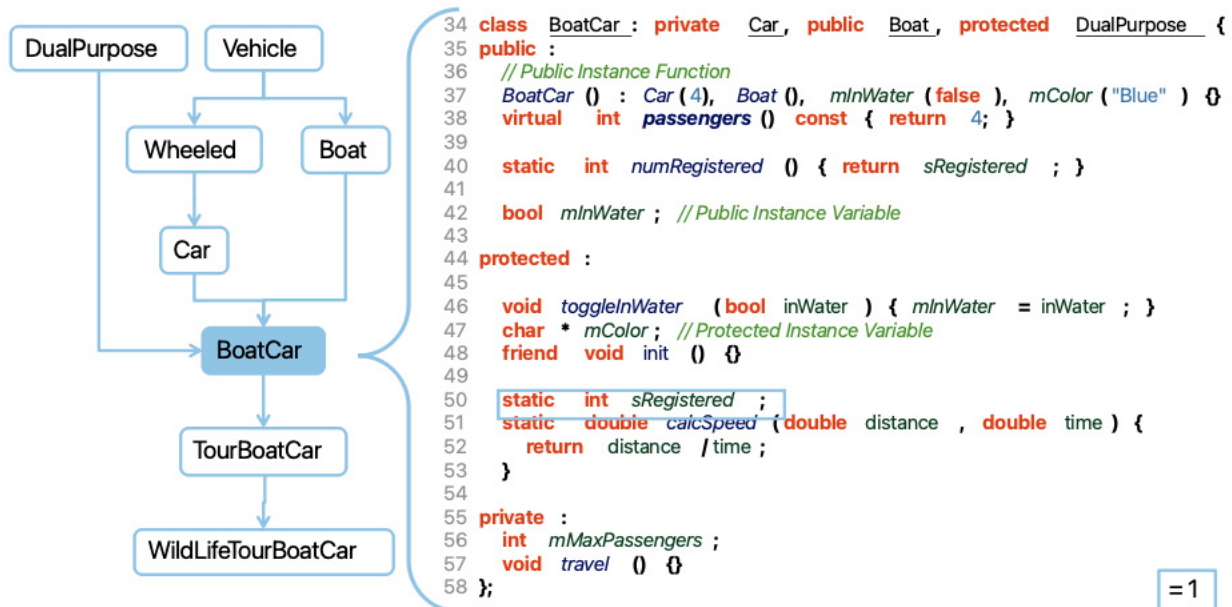
Also Known As: Number of Variables (NV)

Languages: Basic, C#, C++, Java, Pascal, Web

Targets: Architectures, Classes, Files, Project

Number of class variables.

Also known as Lorenz & Kidd - Number of Variables (NV).



Targets By Language:

- **Basic:** Project, Class, Struct
- **C#:** Project, Class, Struct
- **C++:** Project, Class, Struct, Union
- **Java:** Project, File, Class, Interface
- **Pascal:** Project, Class, Interface
- **Web:** Project, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Classes

API ID: CountDeclClass

Languages: Basic, C#, C++, Java, Pascal, Python, Web

Targets: Architectures, Files, Project

Number of classes.

Targets By Language:

- **Basic:** Project, File
- **C#:** Project, File
- **C++:** Project, File
- **Java:** Project, File
- **Pascal:** Project, File
- **Python:** Project, File
- **Web:** Project, File

Configuration:

- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated file metrics" option.
- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show default summary metrics" option.

Code Files

API ID: CountDeclFileCode

Languages: C++

Targets: Architectures, Project

Number of code files.

Targets By Language:

- **C++:** Project

Configuration:

- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated file metrics" option.

Code Lines

API ID: CountLineCode

Also Known As: Lines of Code (LOC), Source Lines of Code (SLOC)

Languages: Ada, Assembly, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, VHDL, Web

Targets: Architectures, Classes, Files, Functions, Modules, Packages, Project,

Subprograms

Number of lines containing source code.

Note that a line can contain source and a comment and thus count towards multiple metrics. For classes this is the sum of CountLineCode for the member functions of the class.

Also known as Lines of Code (LOC); Source Lines of Code (SLOC).

Code	Comment	Preprocessor	Declarative	Executable	Inactive	
✓			✓			11 void SayHello :: printHello () {
✓						12 switch (i) {
✓						13 case 0:
✓						14 cout << "Hello World" << endl ;
✓						15 case 1:
✓						16 cout << "HELLO WORLD!" << endl ;
✓	✓					17 default : //a comment here
✓						18 for (int m = 0; m < j; m++);
✓						19 cout << "hello world" << endl ;
✓						20 }
		✓				21 #ifdef A_VERY_NICE_VARIABLE
					✓	22 cout << "Inactive Line" << endl ; //Inactive
✓	✓				✓	23 #endif
		✓				24
						25
✓						26 }

= Code && not(Inactive)
= 11

Inactive code lines do not count

Targets By Language:

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Project, File, Method, Module, Class
- **C#:** Project, File, Class, Method
- **C++:** Project, File, Class, Struct, Union, Function
- **Fortran:** Project, File, Module, Block Data, Function, Program, Subroutine
- **Java:** Project, File, Class, Interface, Method
- **Jovial:** Project, File, Module, Subroutine
- **Pascal:** Project, File, Class, Interface, Compunit, Function, Procedure
- **Python:** Project, File, Class, Function
- **Rust:** Project, File, Function
- **VHDL:** Project, File, Procedure, Function, Process, Architecture
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- ### Code Lines (Includes Inactive)

Targets: Architectures, Classes, Files, Functions, Project

[illegible]

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.

- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Include inactive lines" option.

Code Lines (JavaScript)

API ID: CountLineCodeJavascript

Languages: Web

Targets: Architectures, Files, Project

Number of JavaScript lines containing source code.

Targets By Language:

- **Web:** Project, File

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Breakdown counts by web language" option.

Code Lines (PHP)

API ID: CountLineCodePhp

Languages: Web

Targets: Architectures, Classes, Files, Project

Number of PHP lines containing source code.

Targets By Language:

- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Breakdown counts by web language" option.

CodeCheck Violation Density by Code Lines

API ID: CCViolDensityCode

Languages: Any

Targets: Architectures, Files, Functions, Project

The density of CodeCheck Violations by Lines of Code

This metric is calculated as [Number of CodeCheck Violations](#) divided by CountLineCode. See also [CodeCheck Violation Density by Lines](#).

CodeCheck Violation Density by Lines

API ID: CCViolDensityLine

Languages: Any

Targets: Architectures, Files, Functions, Project

The density of CodeCheck Violations by Line

This metric is calculated as [Number of CodeCheck Violations](#) divided by CountLine. See also [CodeCheck Violation Density by Code Lines](#).

CodeCheck Violation Types

API ID: CountCCViolType

Languages: Any

Targets: Architectures, Files, Functions, Project

The number of distinct violation types (distinct check ids)

For example, if the violation browser for a file showed these violations:

- non-void function does not return a value
- Remove unused global object `name`
- Remove unused local object dummy
- 'header.h' file not found.

The violation check ids are CPP_WARN_RETURN_TYPE, STI_UNUSED, STI_UNUSED, and UND_ERROR. UND_ERROR is an analysis error and not part of CodeCheck, so the value of this metric is 2 for CPP_WARN_RETURN_TYPE and STI_UNUSED.

See also [CountAnalysisError](#) for a count of analysis errors and [CountAnalysisWarning](#) for a count of analysis warnings.

CodeCheck Violations

API ID: CountCCViol

Languages: Any

Targets: Architectures, Files, Functions, Project

The number of CodeCheck Violations.

For example, if the violation browser for a file showed these violations:

- non-void function does not return a value
- Remove unused global object `name`
- Remove unused local object dummy
- 'header.h' file not found.

Then the value of this metric would be 3 because the fourth violation is an analysis error and this metric only counts CodeCheck violations.

See also [CountAnalysisError](#) for a count of analysis errors and [CountAnalysisWarning](#) for a count of analysis warnings.

Comment Lines

API ID: CountLineComment

Also Known As: Comment Lines of Code (CLOC)

Languages: Ada, Assembly, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, VHDL, Web

Targets: Architectures, Classes, Files, Functions, Modules, Packages, Project, Subprograms

Number of lines containing comment.

This can overlap with other code counting metrics. For instance:

```
int j = 1; // comment
```

has a comment, is a source line, is an executable source line, and is a declarative source line.

Also known as Comment Lines of Code (CLOC).

Code	Comment	Preprocessor	Declarative	Executable	Inactive	
✓						11 void SayHello :: printHello () {
✓						12 switch (i) {
✓						13 case 0:
✓						14 cout << "Hello World" << endl ;
✓						15 case 1:
✓						16 cout << "HELLO WORLD!" << endl ;
✓	✓					17 default : //a comment here
✓						18 for (int m = 0; m < j; m++);
✓						19 cout << "hello world" << endl ;
✓						20 }
		✓				21 #ifdef A_VERY_NICE_VARIABLE
					✓	22
✓	✓				✓	23 cout << "Inactive Line" << endl ; //Inactive
		✓				24 #endif
						25
✓						26 }

= Comment && not(Inactive)
= 1

Inactive comment lines do not count

Targets By Language:

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Project, File, Method, Module, Class
- **C#:** Project, File, Class, Method
- **C++:** Project, File, Class, Struct, Union, Function
- **Fortran:** Project, File, Module, Block Data, Function, Program, Subroutine
- **Java:** Project, File, Class, Interface, Method
- **Jovial:** Project, File, Module, Subroutine
- **Pascal:** Project, File, Class, Interface, Compunit, Function, Procedure
- **Python:** Project, File, Class, Function
- **Rust:** Project, File, Function
- **VHDL:** Project, File, Procedure, Function, Process, Architecture
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Include inactive lines" option for languages that have inactive lines.
- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show default summary metrics" option.

Comment Lines (HTML)

API ID: CountLineCommentHtml**Languages:** Web**Targets:** Architectures, Files, Project

Number of HTML lines containing comment.

Targets By Language:

- **Web:** Project, File

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Breakdown counts by web language" option.

Comment Lines (Includes Inactive)**API ID:** CountLineCommentWithInactive**Languages:** C++**Targets:** Architectures, Classes, Files, Functions, Project

Number of lines containing comment, including inactive regions.

Code	Comment	Preprocessor	Declarative	Executable	Inactive	
✓						11 void SayHello :: printHello () {
✓						12 switch (i) {
✓						13 case 0:
✓						14 cout << "Hello World" << endl ;
✓						15 case 1:
✓						16 cout << "HELLO WORLD!" << endl ;
✓	✓					17 default : //a comment here
✓						18 for (int m = 0; m < j; m++);
✓						19 cout << "hello world" << endl ;
✓						20 }
		✓				21 #ifdef A_VERY_NICE_VARIABLE
						22
✓	✓				✓	23 cout << "Inactive Line" << endl ; //Inactive
		✓				24 #endif
						25
✓						26 }

= Comment
= 2

Targets By Language:

- **C++:** Project, File, Class, Struct, Union, Function

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Include inactive lines" option.

Comment Lines (JavaScript)

API ID: CountLineCommentJavascript

Languages: Web

Targets: Architectures, Files, Project

Number of JavaScript lines containing comment.

Targets By Language:

- **Web:** Project, File

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Breakdown counts by web language" option.

Comment Lines (PHP)

API ID: CountLineCommentPhp

Languages: Web

Targets: Architectures, Classes, Files, Project

Number of PHP lines containing comment.

Targets By Language:

- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and

Statements" with the "Show line count metrics" option.

- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Breakdown counts by web language" option.

Comment to Code Ratio

API ID: RatioCommentToCode

Languages: Ada, Assembly, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, VHDL, Web

Targets: Architectures, Classes, Files, Functions, Modules, Packages, Project, Subprograms

Ratio of comment lines to code lines.

Note that because some lines are both code and comment, this could easily yield percentages higher than 100.

Comment
Code

```

✓ 11 void SayHello :: printHello () {
✓ 12     switch (i) {
✓ 13         case 0:
✓ 14             cout << "Hello World" << endl ;
✓ 15         case 1:
✓ 16             cout << "HELLO WORLD!" << endl ;
✓ 17         default : //a comment here
✓ 18             for (int m = 0; m < j; m++);
✓ 19             cout << "hello world" << endl ;
✓ 20     }
21     #ifdef A_VERY_NICE_VARIABLE
22
23         cout << "Inactive Line" << endl ; //inactive
24     #endif
25
✓ 26 }
```

$$= \text{CountLineComment} / \text{CodeLineCode}$$

$$= 1 / 11$$

$$= 0.09$$

Targets By Language:

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Project, File, Method, Module, Class
- **C#:** Project, File, Class, Method
- **C++:** Project, File, Class, Struct, Union, Function
- **Fortran:** Project, File, Block Data, Function, Program, Subroutine
- **Java:** Project, File, Class, Interface, Method

- **Jovial:** Project, File, Module, Subroutine
- **Pascal:** Project, File, Class, Interface, Compunit, Function, Procedure
- **Python:** Project, File, Class, Function
- **Rust:** Project, File, Function
- **VHDL:** Project, File, Procedure, Function, Architecture
- **Web:** Project, File

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show default summary metrics" option.

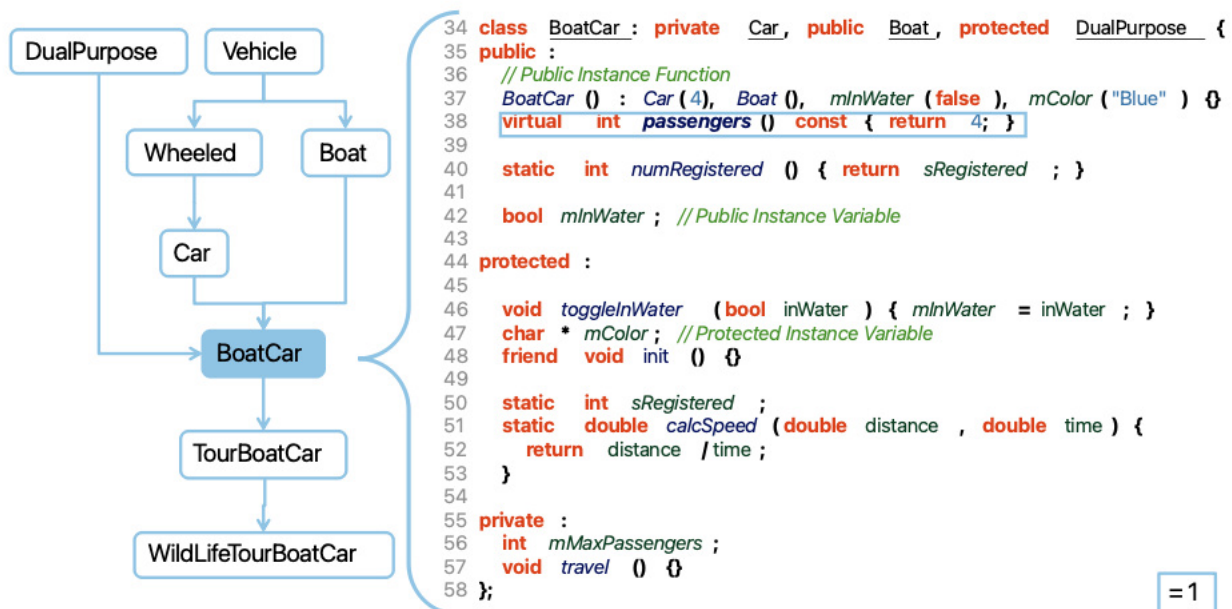
Const Methods

API ID: CountDeclMethodConst

Languages: C++

Targets: Architectures, Classes, Project

Number of local const methods.



Targets By Language:

- **C++:** Project, Class, Struct, Union

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Coupled Classes

API ID: CountClassCoupled

Also Known As: Coupling Between Objects (CBO)

Languages: Basic, C#, C++, Java, Pascal, Python

Targets: Classes

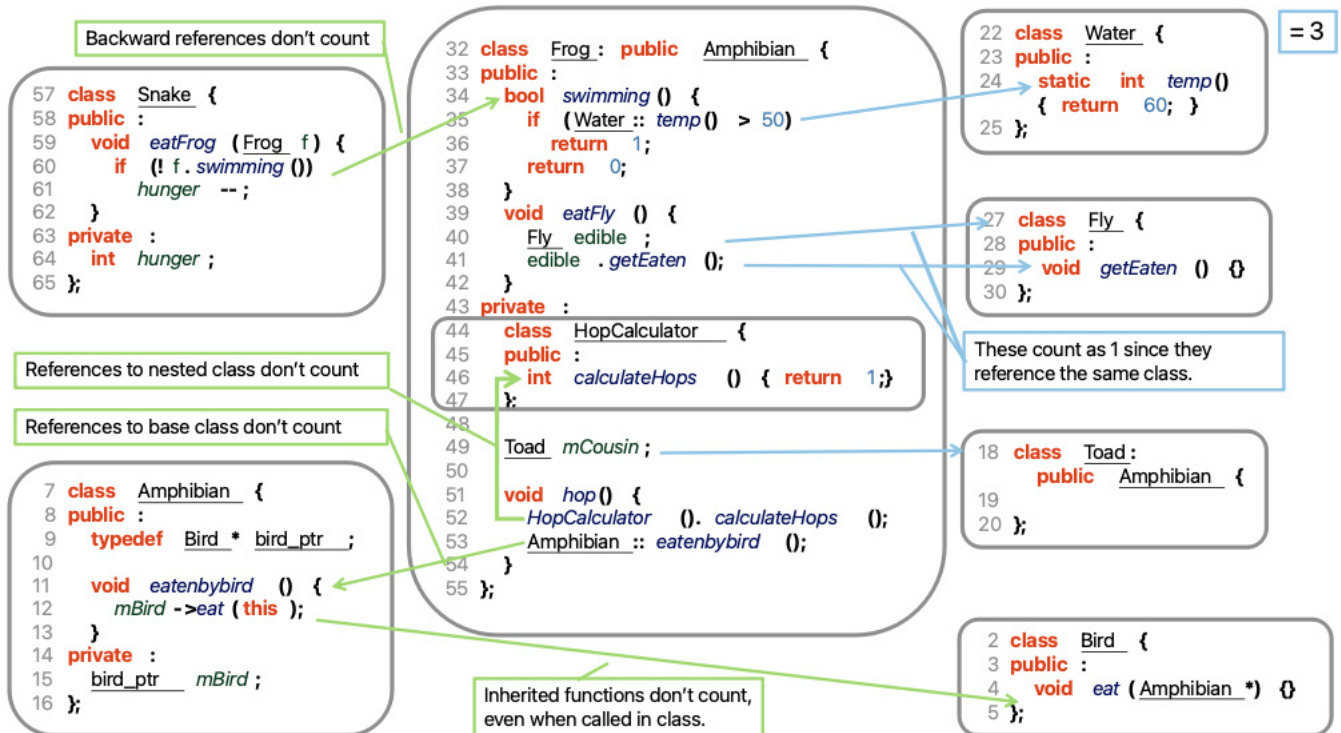
Number of other classes coupled to.

The Coupling Between Object Classes (CBO) measure for a class is a count of the number of other classes to which it is coupled. Base classes and nested classes are not counted. Class A is coupled to class B if class A uses a type, data, or member from class B. This metric is also referred to as Efferent Coupling (Ce). Any number of couplings to a given class counts as 1 towards the metric total.

Chidamber & Kemerer suggest that:

- 1) Excessive coupling between object classes is detrimental to modular design and prevents reuse.
- 2) Inter-object class couples should be kept to a minimum.
- 3) The higher the inter-object class coupling, the more rigorous testing needs to be.

Also known as Chidamber & Kemerer - Coupling Between Objects (CBO).



Targets By Language:

- **Basic:** Class, Struct
- **C#:** Class, Struct
- **C++:** Class, Struct, Union
- **Java:** Class, Interface
- **Pascal:** Class, Interface
- **Python:** Class

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show coupling and cohesion metrics" option.

Coupled Classes Modified

API ID: CountClassCoupledModified

Languages: Basic, C#, Java, Pascal, Python

Targets: Classes

Number of other non-standard classes coupled to.

Targets By Language:

- **Basic:** Class, Struct
- **C#:** Class, Struct
- **Java:** Class, Interface

- **Pascal:** Class, Interface
- **Python:** Class

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show coupling and cohesion metrics" option.

Coupled Packages

API ID: CountPackageCoupled

Languages: Ada

Targets: Packages

Number of other packages coupled to.

Targets By Language:

- **Ada:** Package

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show coupling and cohesion metrics" option.

Cyclomatic Complexity

API ID: Cyclomatic

Also Known As: Cyclomatic Complexity (CC)

Languages: Ada, Assembly, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, VHDL, Web

Targets: Files, Functions, Modules, Packages, Subprograms

McCabe Cyclomatic Complexity, the number of decision points + 1.

McCabe Cyclomatic complexity as per the original NIST paper on the subject. The cyclomatic complexity of any structured program with only one entrance point and one exit point is equal to the number of decision points contained in that program plus one. Understand counts the keywords for decision points (FOR, WHILE, etc) and then adds 1. For a switch statement, each 'case' is counted as 1. For languages with macros, the expanded macro text is also included in the calculation.

Also known as McCabe - Cyclomatic Complexity (CC).



Targets By Language:

- **Ada:** Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Method
- **C#:** Method
- **C++:** Function
- **Fortran:** Module, Block Data, Function, Program, Subroutine
- **Java:** Method
- **Jovial:** Subroutine
- **Pascal:** Compunit, Function, Procedure
- **Python:** File, Function
- **Rust:** Function
- **VHDL:** Procedure, Function, Process
- **Web:** File

Configuration:

- The cyclomatic metric used is configured from "Project Configuration/ Metrics/Complexity". This metric results from having "Count each decision in a multi-way-decision structure" on and having "Count logical conjunctions (OR) and logical and (AND)" off. Checking "Show all available cyclomatic complexity metric variants" will also enable the metric.

Declarative Code Lines

API ID: CountLineCodeDecl

Languages: Ada, Assembly, C#, C++, Fortran, Java, Pascal, Python

Number of lines containing declarative source code. Note that a line can be declarative and executable - "int i =0;" for instance.

= Code && Declarative
= 2

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **C#:** Project, File, Class, Method
- **C++:** Project, File, Class, Struct, Union, Function
- **Fortran:** Project, File, Module, Block Data, Function, Program, Subroutine
- **Java:** Project, File, Class, Interface, Method
- **Pascal:** Project, File, Class, Interface, Compunit, Function, Procedure
- **Python:** Project, File, Class, Function

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.

Languages: Ada, Assembly, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python,

VHDL, Web

Targets: Architectures, Classes, Files, Functions, Modules, Packages, Project, Subprograms

Number of declarative statements.

	Executive	Declarative	Empty	Total
60 int main () {	0	1	0	1
61 for (int i = 0;	1	0	0	1
62 i < 10;	1	0	0	1
63 i ++)	0	0	1	1
64 ;				
65				
66 int j = func 0;	1 (0)	1	0	1
67 int k = 0;	0	1	0	1
68 int l = 1;	0	1	0	1
69				
70 int m	0	1	0	1
71 = func 0;	1 (0)	0	0	0
72 int n;	0	1	0	1
73 n = 1;	1	0	0	1
74				
75 #ifdef A_VERY_NICE_VARIABLE				
76 int j = 0;				
77 cout << j << endl ;				
78				
79 #endif				
80				
81 return 0;	1	0	0	1
82 }	0	0	0	0
	6 (4)	6	1	11

Statement is declarative, but only counts at file scope

Initializations with function calls are both declarative and executable in strict

Inactive code is not counted

Fuzzy values that differ from strict values are shown in parentheses.

Targets By Language:

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Project, File, Method, Module, Class
- **C#:** Project, File, Class, Method
- **C++:** Project, File, Class, Struct, Union, Function
- **Fortran:** Project, File, Module, Block Data, Function, Program, Subroutine
- **Java:** Project, File, Class, Interface, Method
- **Jovial:** Project, File, Module, Subroutine
- **Pascal:** Project, File, Class, Interface, Compunit, Function, Procedure
- **Python:** Project, File, Class, Function
- **VHDL:** Project, File
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show statement count metrics" option.
- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show default

summary metrics" option.

Declarative Statements (Javascript)

API ID: CountStmtDeclJavascript

Languages: Web

Targets: Architectures, Files, Project

Number of JavaScript declarative statements.

Targets By Language:

- **Web:** Project, File

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show statement count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Breakdown counts by web language" option.

Declarative Statements (PHP)

API ID: CountStmtDeclPhp

Languages: Web

Targets: Architectures, Classes, Files, Project

Number of PHP declarative statements.

Targets By Language:

- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show statement count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Breakdown counts by web language" option.

Default Methods

API ID: CountDeclMethodDefault

Languages: Java

Targets: Architectures, Classes, Files, Project

Number of local default methods.

Targets By Language:

- **Java:** Project, File, Class, Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Derived Classes

API ID: CountClassDerived

Also Known As: Number of Children (NOC)

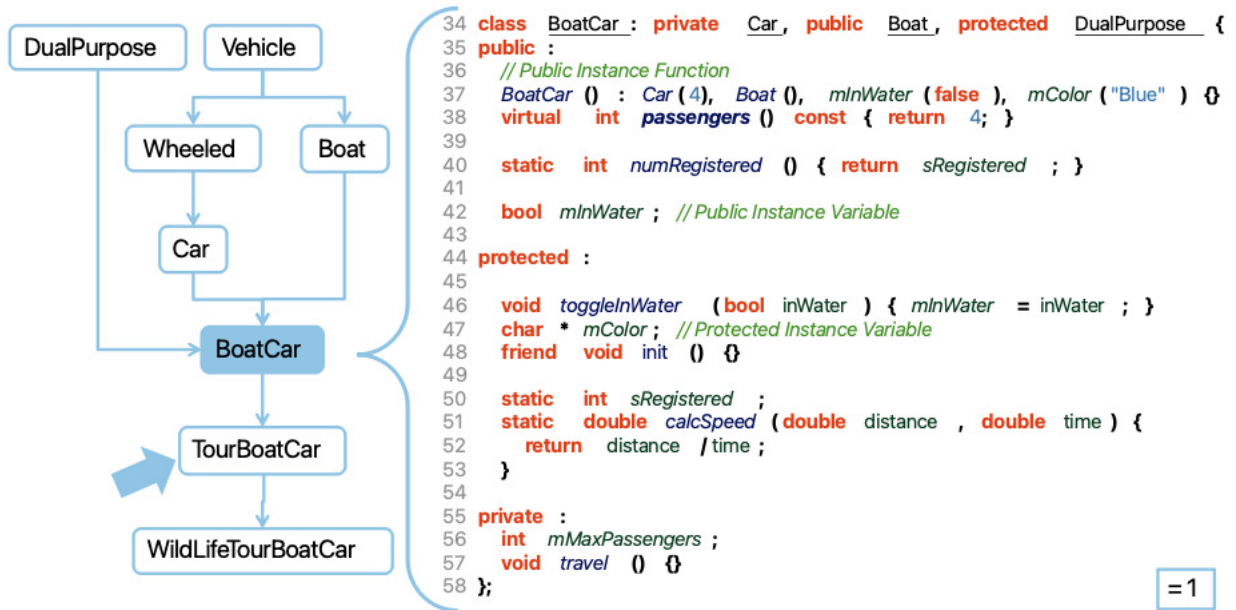
Languages: Basic, C#, C++, Java, Pascal, Python, Web

Targets: Classes

Number of immediate subclasses.

(i.e. the number of classes one level down the inheritance tree from this class).

Also known as Chidamber & Kemerer - Number of Children (NOC).



= 1

Targets By Language:

- **Basic:** Class, Struct
- **C#:** Class, Struct
- **C++:** Class, Struct, Union
- **Java:** Class, Interface
- **Pascal:** Class, Interface
- **Python:** Class
- **Web:** PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show inheritance metrics" option.

Empty Statements

API ID: CountStmtEmpty

Languages: C++

Targets: Architectures, Classes, Files, Functions, Project

Number of empty statements.

Statement is declarative, but only counts at file scope	<pre> 60 int main () { 61 for (int i = 0; 62 i < 10; 63 i++) 64 ; 65 66 int j = func (); 67 int k = 0; 68 int l = 1; 69 70 int m 71 = func (); 72 int n; 73 n = 1; 74 75 #ifdef A_VERY_NICE_VARIABLE 76 77 int j = 0; 78 cout << j << endl ; 79 #endif 80 81 return 0; 82 } </pre>	<table> <tr> <th>Executive</th> <th>Declarative</th> <th>Empty</th> <th>Total</th> </tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>1 (0)</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1 (0)</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td></tr> </table>	Executive	Declarative	Empty	Total	0	1	0	1	1	0	0	1	1	0	0	1	0	0	1	1	1 (0)	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1 (0)	0	0	0	0	1	0	1	1	0	0	1
Executive	Declarative	Empty	Total																																															
0	1	0	1																																															
1	0	0	1																																															
1	0	0	1																																															
0	0	1	1																																															
1 (0)	1	0	1																																															
0	1	0	1																																															
0	1	0	1																																															
0	1	0	1																																															
1 (0)	0	0	0																																															
0	1	0	1																																															
1	0	0	1																																															
Initializations with function calls are both declarative and executable in strict		Fuzzy values that differ from strict values are shown in parentheses.																																																
Inactive code is not counted																																																		
		<table> <tr> <td>1</td> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>6 (4)</td> <td>6</td> <td>1</td> <td>11</td> </tr> </table>	1	0	0	1	0	0	0	0	6 (4)	6	1	11																																				
1	0	0	1																																															
0	0	0	0																																															
6 (4)	6	1	11																																															

Targets By Language:

- **C++:** Project, File, Class, Struct, Union, Function

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show statement count metrics" option.

Essential Complexity

API ID: Essential

Also Known As: ev(G)

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Files, Functions, Modules, Packages, Subprograms

The number of decision points + 1 after control graph reduction.

Essential complexity is the cyclomatic complexity after iteratively replacing all well structured control structures with a single statement. Structures such as if-then-else and while loops are considered well structured. Understand calculates the essential complexity by removing all the structured subgraphs from the control graph and then calculating the complexity. A graph that has only the regular single entry/single exit loops or branches will be reducible to a graph with complexity one. Any branches into or out of a loop or decision will make the graph non-reducible and will have Essential Complexity > 2. (You never get 2 since a graph with complexity 2

is always reducible to a graph with complexity 1)

Also known as McCabe - $ev(G)$.

```

4 void knotsDemo () {
5   while (1) {
6     if (a)
7       break ;
8     if (b || c) {
9       if (d || e) {
10
11       }
12     }
13     else {
14       if (i)
15         dosomething 0;
16       else if (j)
17         dosomething 0;
18       else if (k)
19         dosomething 0;
20       else {}
21     }
22   }
23 }
24 }

```

Reduction

```

void knotsDemo () {
  while (1) {
    if (a)
      break ;
  }
}

```

= Cyclomatic
 = decision points + 1
 = **while** + **if** + 1
 = 3

Targets By Language:

- **Ada:** Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Method
- **C#:** Method
- **C++:** Function
- **Fortran:** Module, Block Data, Function, Program, Subroutine
- **Java:** Method
- **Jovial:** Subroutine
- **Pascal:** Compunit, Function, Procedure
- **Python:** File, Function
- **Rust:** File, Function
- **Web:** File

Configuration:

- This metric can be enabled from "Project Configuration/Metrics/Complexity" with the "Essential" option.

Executable Code Lines

API ID: CountLineCodeExe

Languages: Ada, Assembly, C#, C++, Fortran, Java, Pascal, Python

Targets: Architectures, Classes, Files, Functions, Modules, Packages, Project, Subprograms

Number of lines containing executable source code.

[illegible]

Targets By Language:

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **C#:** Project, File, Class, Method
- **C++:** Project, File, Class, Struct, Union, Function
- **Fortran:** Project, File, Module, Block Data, Function, Program, Subroutine
- **Java:** Project, File, Class, Interface, Method
- **Pascal:** Project, File, Class, Interface, Compunit, Function, Procedure
- **Python:** Project, File, Class, Function

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.

Executable Statements

API ID: CountStmtExe

Languages: Ada, Assembly, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python,

VHDL, Web

Targets: Architectures, Classes, Files, Functions, Modules, Packages, Project, Subprograms

Number of executable statements.

Statement is declarative, but only counts at file scope	<pre> 60 int main () { 61 for (int i = 0; 62 i < 10; 63 i++) 64 ; 65 66 int j = func (); 67 int k = 0; 68 int l = 1; 69 70 int m 71 = func (); 72 int n; 73 n = 1; 74 75 #ifdef A_VERY_NICE_VARIABLE 76 int j = 0; 77 cout << j << endl ; 78 #endif 79 80 81 return 0; 82 } </pre>	<table> <tr> <th>Executive</th> <th>Declarative</th> <th>Empty</th> <th>Total</th> </tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>1 (0)</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1 (0)</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td></tr> </table>	Executive	Declarative	Empty	Total	0	1	0	1	1	0	0	1	1	0	0	1	0	0	1	1	1 (0)	1	0	1	0	1	0	1	0	1	0	1	1 (0)	0	0	0	0	1	0	1	1	0	0	1
Executive	Declarative	Empty	Total																																											
0	1	0	1																																											
1	0	0	1																																											
1	0	0	1																																											
0	0	1	1																																											
1 (0)	1	0	1																																											
0	1	0	1																																											
0	1	0	1																																											
1 (0)	0	0	0																																											
0	1	0	1																																											
1	0	0	1																																											
Initializations with function calls are both declarative and executable in strict																																														
Inactive code is not counted		Fuzzy values that differ from strict values are shown in parentheses.																																												
		<table> <tr> <td>1</td> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>6 (4)</td> <td>6</td> <td>1</td> <td>11</td> </tr> </table>	1	0	0	1	0	0	0	0	6 (4)	6	1	11																																
1	0	0	1																																											
0	0	0	0																																											
6 (4)	6	1	11																																											

Targets By Language:

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Project, File, Method, Module, Class
- **C#:** Project, File, Class, Method
- **C++:** Project, File, Class, Struct, Union, Function
- **Fortran:** Project, File, Module, Block Data, Function, Program, Subroutine
- **Java:** Project, File, Class, Interface, Method
- **Jovial:** Project, File, Subroutine
- **Pascal:** Project, File, Class, Interface, Compunit, Function, Procedure
- **Python:** Project, File, Class, Function
- **VHDL:** Project, File
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show statement count metrics" option.
- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show default

summary metrics" option.

Executable Statements (JavaScript)

API ID: CountStmtExeJavascript

Languages: Web

Targets: Architectures, Files, Project

Number of JavaScript executable statements.

Targets By Language:

- **Web:** Project, File

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show statement count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Breakdown counts by web language" option.

Executable Statements (PHP)

API ID: CountStmtExePhp

Languages: Web

Targets: Architectures, Classes, Files, Project

Number of PHP executable statements.

Targets By Language:

- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show statement count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Breakdown counts by web language" option.

Executable Units

API ID: CountDeclExecutableUnit

Languages: Ada, Basic, C#, Fortran, Java, Pascal, Python, Web

Targets: Architectures, Files, Project

Number of program units with executable code.

Targets By Language:

- **Ada:** Project, File
- **Basic:** Project, File
- **C#:** Project, File
- **Fortran:** Project, File
- **Java:** Project, File
- **Pascal:** Project, File
- **Python:** Project, File
- **Web:** Project, File

Configuration:

- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated file metrics" option.

Files

API ID: CountDeclFile

Languages: Ada, Assembly, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, VHDL, Web

Targets: Architectures, Project

Number of files.

Targets By Language:

- **Ada:** Project
- **Basic:** Project
- **C#:** Project
- **C++:** Project
- **Fortran:** Project
- **Java:** Project
- **Jovial:** Project
- **Pascal:** Project
- **Python:** Project
- **Rust:** Project
- **VHDL:** Project
- **Web:** Project

Configuration:

- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated file metrics" option.
- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show default summary metrics" option.

Friend Methods**API ID:** CountDeclMethodFriend**Also Known As:** Number of Friend Methods (NFM), Number of Friends (NF)**Languages:** C++**Targets:** Classes

Number of local (not inherited) friend methods.

The number of friend functions plus the CountDeclMethod of friend classes.

Also known as Lorenz & Kidd - Number of Friends (NF); Number of Friend Methods (NFM).

```

1 class CohesionClass {
2 public :
3     void func1 () {
4         ...
5     }
6
7     void func2 () {
8         mVar1 = 4;
9     }
10
11     static void addObj () {
12         sNumObjs ++;
13     }
14 protected :
15
16     void func3 () {
17         mVar2 = "blue" ;
18     }
19 private :
20
21     void func4 () {
22         ...
23     }
24
25     int mVar1 ;
26     char * mVar2 ;
27     static int sNumObjs ;
28 };
  
```

```

1 class FriendDemo {
2     friend class CohesionClass ;
3
4     friend void init () ;
5
6 };
  
```

= number of friend functions +
CountDeclMethod of friend classes
= 1 + 5
= 6

Targets By Language:

- **C++:** Class, Struct, Union

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Functions

API ID: CountDeclFunction

Languages: Assembly, C#, C++, Java, Python, Rust, Web

Targets: Architectures, Files, Project

Number of functions.

Targets By Language:

- **C#:** Project, File
- **C++:** Project, File
- **Java:** Project, File
- **Python:** Project, File
- **Rust:** Project, File
- **Web:** Project, File

Configuration:

- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated file metrics" option.
- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show default summary metrics" option.

Header Files

API ID: CountDeclFileHeader

Languages: C++

Targets: Architectures, Project

Number of header files.

Targets By Language:

- **C++:** Project

Configuration:

- ## Inactive Lines

Targets: Architectures, Classes, Files, Functions, Modules, Project, Subprograms

This is the number of lines that are inactive from the view of the preprocessor. In other words, they are on the FALSE side of a `#if` or `#ifdef` preprocessor directive.

[illegible]

- **C#**: Project, File, Class, Method
- **C++**: Project, File, Class, Struct, Union, Function
- **Pascal**: Project, File, Class, Interface, Compunit, Function, Procedure

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- For projects and architectures, this metric is enabled from "Project

Configuration/Metrics/Projects and Architectures" with the "Show default summary metrics" option.

Inputs

API ID: CountInput

Also Known As: FANIN (Infomational fan-in)

Languages: C#, C++, Fortran, Java

Targets: Functions, Subprograms

Number of calling subprograms plus global variables read.

The number of inputs a function uses plus the number of unique subprograms calling the function. Inputs include parameters and global variables that are used in the function, so Functions calledby + Parameters read + Global Variables read. Recursive function calls and local variables that are not class static variables are not included. Of the two general approaches to calculating FANIN (informational versus structural) ours is the informational approach.

Also known as FANIN (Infomational fan-in).

```

3  int in = 1;
4  int out = 1;
5
6  int inOutFunc (int in1 , int in2 , int *inout1 , int &inout2 , int * out1 , int & out2 ) {
7      out = in + in1 + in2 + *inout1 + inout2 ;
8
9      *inout1 = in1 ;
10     inout2 = in2 ;
11
12     *out1 = in1 ;
13     out2 = in2 ;
14
15     in1 = somefunc ();
16     in2 = 2;
17
18     int randomint = 3;
19     in1 = randomint ;
20
21     return 4;
22 }
...
24 void callingfunc () {
25     int a, b, c, d;
26     int myval = inOutFunc (1, 2, &a, b, &c, d);

```

= functions called -by + parameters used + globals used
= 1 + 4 + 1
= 6

Entity	Counts?	Comment
in	Yes	Use line 7
out	No	Not used
in1	Yes	Use line 7, Use line 9, Use line 12
in2	Yes	Use line 7, Use line 10, Use line 13
inout1	Yes	Use line 7
inout2	Yes	Use line 7
out1	No	Not used
out2	No	Not used
randomint	No	Not a parameter, global, or class static variable
calledbyfunc	Yes	Line 26

Targets By Language:

- **C#:** Method
- **C++:** Function

- **Fortran:** Function, Program, Subroutine
- **Java:** Method

Instance Methods

API ID: CountDeclInstanceMethod

Also Known As: Number of Instance Methods (NIM)

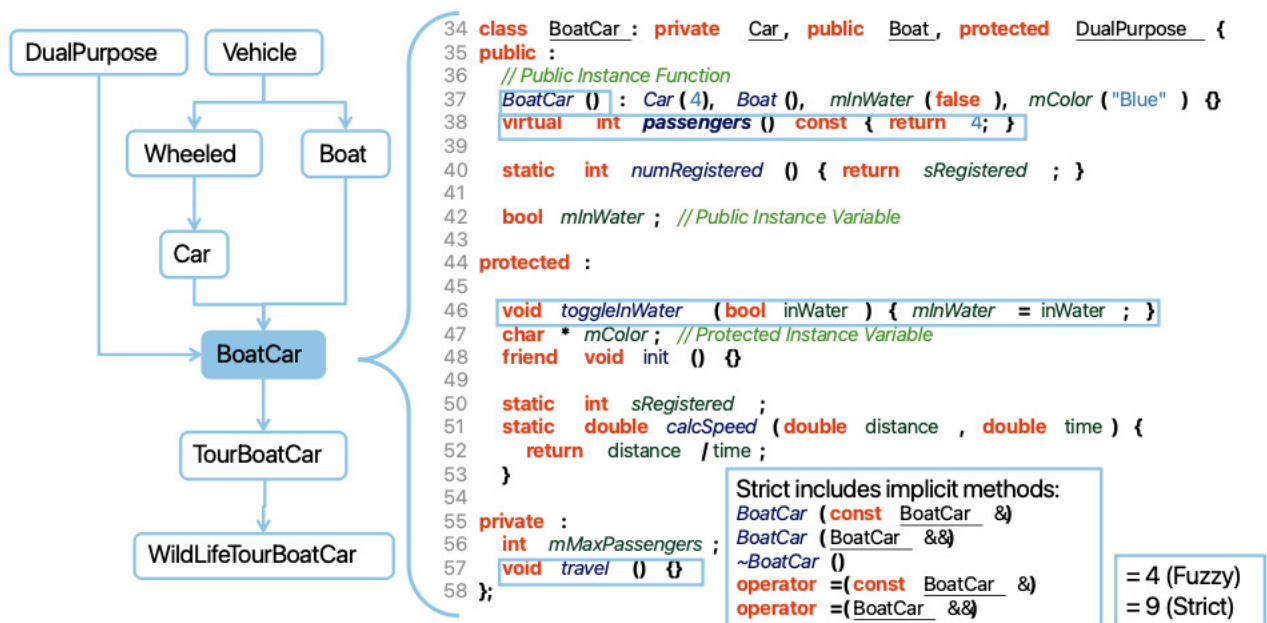
Languages: Basic, C#, C++, Java, Pascal, Python, Web

Targets: Architectures, Classes, Files, Project

Number of instance methods.

Methods defined in a class that are only accessible through an object of that class.

Also known as Number of Instance Methods (NIM).



Targets By Language:

- **Basic:** Project, Class, Struct
- **C#:** Project, Class, Struct
- **C++:** Project, Class, Struct, Union
- **Java:** Project, File, Class, Interface
- **Pascal:** Project, Class, Interface
- **Python:** Project, Class
- **Web:** Project, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Instance Variables

API ID: CountDeclInstanceVariable

Also Known As: Number of Instance Variables (NIV)

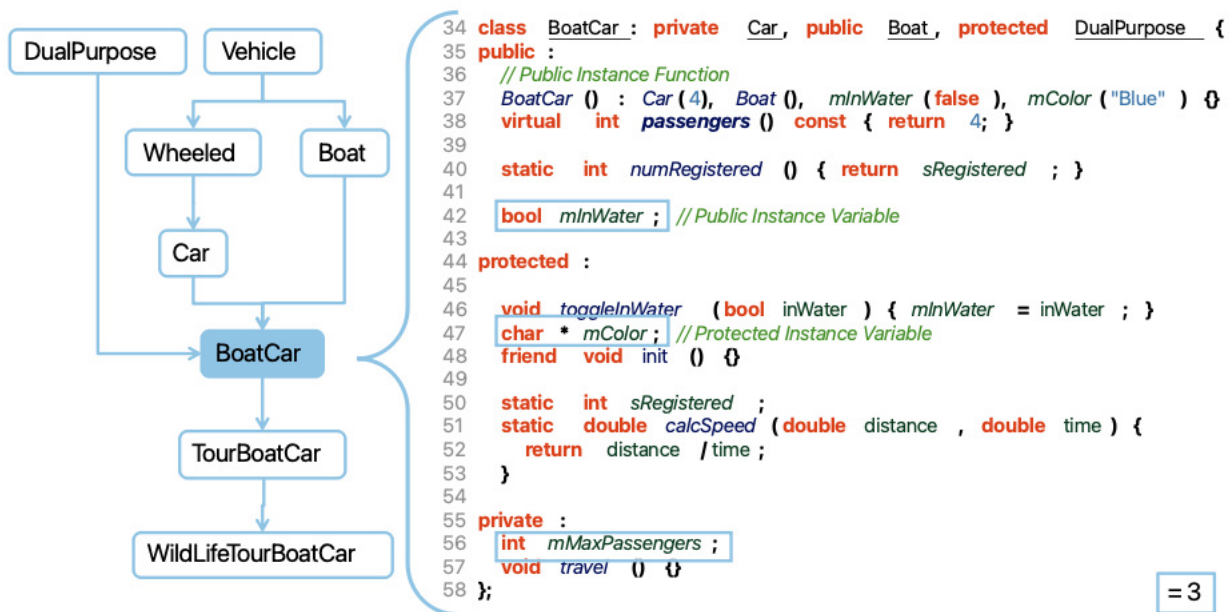
Languages: Basic, C#, C++, Java, Pascal, Python, Web

Targets: Architectures, Classes, Files, Project

Number of instance variables.

Variables defined in a class that are only accessible through an object of that class.

Also known as Number of Instance Variables (NIV).



Targets By Language:

- **Basic:** Project, Class, Struct
- **C#:** Project, Class, Struct
- **C++:** Project, Class, Struct, Union

- **Java:** Project, File, Class, Interface
- **Pascal:** Project, Class, Interface
- **Python:** Project, Class
- **Web:** Project, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Internal Instance Variables

API ID: CountDeclInstanceVariableInternal

Languages: C#

Targets: Architectures, Classes, Project

Number of internal instance variables.

Targets By Language:

- **C#:** Project, Class, Struct

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Internal Methods

API ID: CountDeclMethodInternal

Languages: C#

Targets: Architectures, Classes, Project

Number of local internal methods.

Targets By Language:

- **C#:** Project, Class, Struct

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Knots

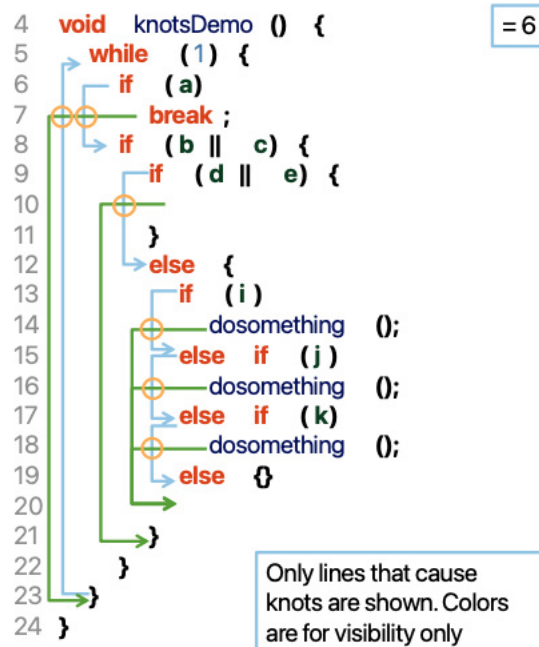
API ID: Knots

Languages: Ada, C#, C++, Java, Rust

Targets: Functions, Packages, Subprograms

Measure of overlapping jumps.

If a piece of code has arrowed lines indicating where every jump in the flow of control occurs, a knot is defined as where two such lines cross each other. The number of knots is proportional to the complexity of the control flow.



Targets By Language:

- **Ada:** Type, Entry, Function, Package, Procedure, Protected, Task
- **C#:** Method
- **C++:** Function

- **Java:** Method
- **Rust:** Function

Configuration:

- This metric can be enabled from "Project Configuration/Metrics/Complexity" with the "Knots" option.

Lines

API ID: CountLine

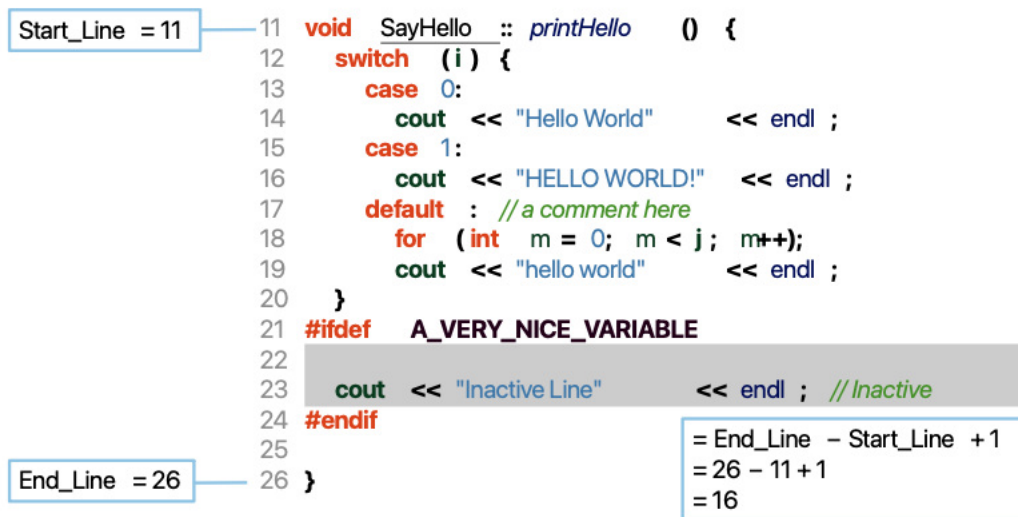
Also Known As: Number of Lines (NL)

Languages: Ada, Assembly, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, VHDL, Web

Targets: Architectures, Classes, Files, Functions, Modules, Packages, Project, Subprograms

Number of physical lines.

Also known as Number of Lines (NL).



```

11 void SayHello :: printHello 0 {
12     switch (i) {
13         case 0:
14             cout << "Hello World" << endl ;
15         case 1:
16             cout << "HELLO WORLD!" << endl ;
17         default : //a comment here
18             for (int m = 0; m < j; m++);
19             cout << "hello world" << endl ;
20     }
21     #ifdef A_VERY_NICE_VARIABLE
22         cout << "Inactive Line" << endl ; //inactive
23     #endif
24 }
25
26

```

Start_Line = 11

End_Line = 26

$$= \text{End_Line} - \text{Start_Line} + 1$$

$$= 26 - 11 + 1$$

$$= 16$$

Targets By Language:

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Project, File, Method, Module, Class
- **C#:** Project, File, Class, Method

- **C++:** Project, File, Class, Struct, Union, Function
- **Fortran:** Project, File, Module, Block Data, Function, Program, Subroutine
- **Java:** Project, File, Class, Interface, Method
- **Jovial:** Project, File, Module, Subroutine
- **Pascal:** Project, File, Class, Interface, Compunit, Function, Procedure
- **Python:** Project, File, Class, Function
- **Rust:** Project, File, Function
- **VHDL:** Project, File, Procedure, Function, Process, Architecture
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show default summary metrics" option.

Lines (HTML)

API ID: CountLineHtml

Languages: Web

Targets: Architectures, Files, Project

Number of all HTML lines.

Targets By Language:

- **Web:** Project, File

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Breakdown counts by web language" option.

Lines (JavaScript)

API ID: CountLineJavascript

Languages: Web

Targets: Architectures, Files, Project

Number of all JavaScript lines.

Targets By Language:

- **Web:** Project, File

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Breakdown counts by web language" option.

Lines (PHP)

API ID: CountLinePhp

Languages: Web

Targets: Architectures, Classes, Files, Project

Number of all PHP lines.

Targets By Language:

- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Lines and Statements" with the "Breakdown counts by web language" option.

Max Cyclomatic Complexity

API ID: MaxCyclomatic

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Architectures, Classes, Files, Modules, Packages, Project

Maximum cyclomatic complexity of all nested functions or methods.

```

1 #include <iostream>
2 using namespace std ;
3
4 class SayHello {
5 public :
6     SayHello () {}
7     void printHello () ;
8 };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {
...
56 }
...
59 int func () ;
60 int main () {
...
82 }
83

```

Annotations for Cyclomatic Complexity:

- `SayHello () {}`: = 1
- `void printHello () ;`: = 4
- `void cyclomaticDemo () {`: = 10
- `int func () ;`: = 2

Class : SayHello
 = Max(1,4)
 = 4

File : sample.cpp
 = Max(1,4,10,2)
 = 10

func is declared here, not defined,
 so it does not count towards file max

Targets By Language:

- **Ada:** Project, File, Package
- **Basic:** Project, File, Module, Class, Struct
- **C#:** Project, File, Class, Struct
- **C++:** Project, File, Class, Struct, Union
- **Fortran:** Project, File
- **Java:** Project, File, Class, Interface
- **Jovial:** Project, File
- **Pascal:** Project, File, Class, Interface, Compunit
- **Python:** Project, File, Class
- **Rust:** Project, File
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- The cyclomatic metric used is configured from "Project Configuration/Metrics/Complexity". This metric results from having "Count each decision in a multi-way-decision structure" on and having "Count logical conjunctions (OR) and logical and (AND)" off. Checking "Show all available cyclomatic complexity metric variants" will also enable the metric.

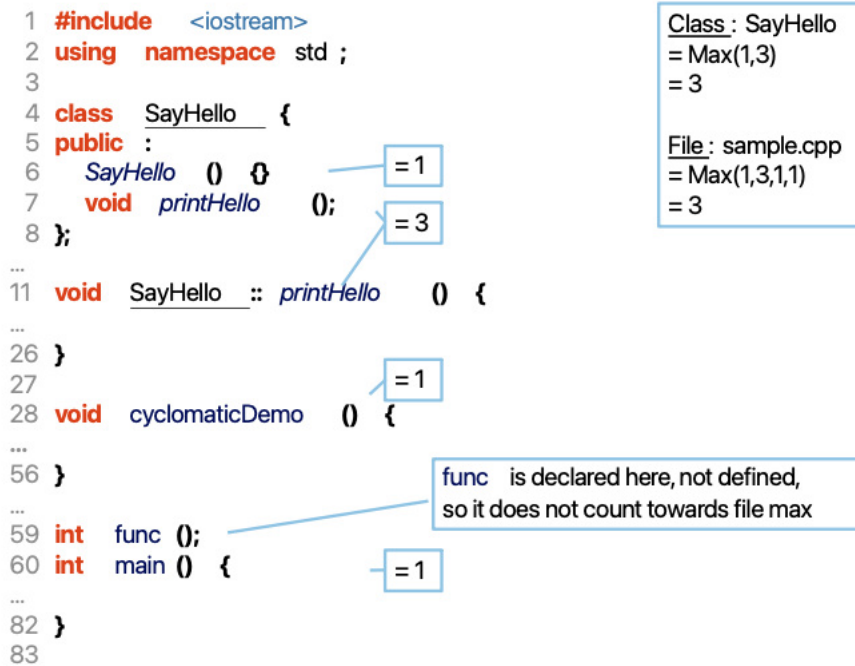
Max Essential Complexity

API ID: MaxEssential

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Architectures, Classes, Files, Modules, Packages, Project

Maximum essential complexity of all nested functions or methods.



```

1  #include <iostream>
2  using namespace std ;
3
4  class SayHello {
5  public :
6      SayHello () {}
7      void printHello () ;
8  };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {
...
56 }
...
59 int func () ;
60 int main () {
...
82 }
83

```

Annotations in the image:

- Line 6: `SayHello () {}` → = 1
- Line 7: `void printHello () ;` → = 3
- Line 28: `void cyclomaticDemo () {` → = 1
- Line 59: `int func () ;` → = 1

Summary boxes:

- Class : SayHello**
= Max(1,3)
= 3
- File : sample.cpp**
= Max(1,3,1,1)
= 3
- func** is declared here, not defined, so it does not count towards file max

Targets By Language:

- **Ada:** Project, File, Package
- **Basic:** Project, File, Module, Class, Struct
- **C#:** Project, File, Class, Struct
- **C++:** Project, File, Class, Struct, Union
- **Fortran:** Project, File
- **Java:** Project, File, Class, Interface
- **Jovial:** Project, File
- **Pascal:** Project, File, Class, Interface, Compunit
- **Python:** Project, File, Class
- **Rust:** Project, File
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Complexity"

with the "Essential" option.

Max Essential Knots

API ID: MaxEssentialKnots

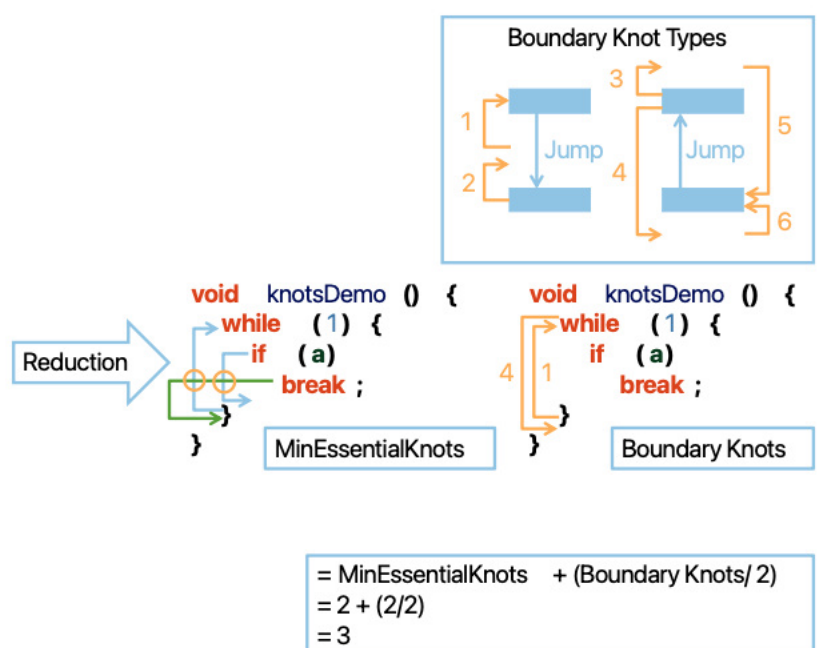
Languages: Ada, C#, C++, Java, Rust

Targets: Functions, Packages, Subprograms

Maximum Knots after structured programming constructs have been removed.

```

4  void knotsDemo () {
5      while (1) {
6          if (a)
7              break ;
8          if (b || c) {
9              if (d || e) {
10
11              }
12          } else {
13              if (i)
14                  dosomething 0;
15              else if (j)
16                  dosomething 0;
17              else if (k)
18                  dosomething 0;
19              else {}
20
21          }
22      }
23  }
24 }
```



Targets By Language:

- **Ada:** Type, Entry, Function, Package, Procedure, Protected, Task
- **C#:** Method
- **C++:** Function
- **Java:** Method
- **Rust:** Function

Configuration:

- This metric can be enabled from "Project Configuration/Metrics/Complexity" with the "Knots" option.

Max Inheritance Tree

API ID: MaxInheritanceTree

Also Known As: Depth of Inheritance Tree (DIT)

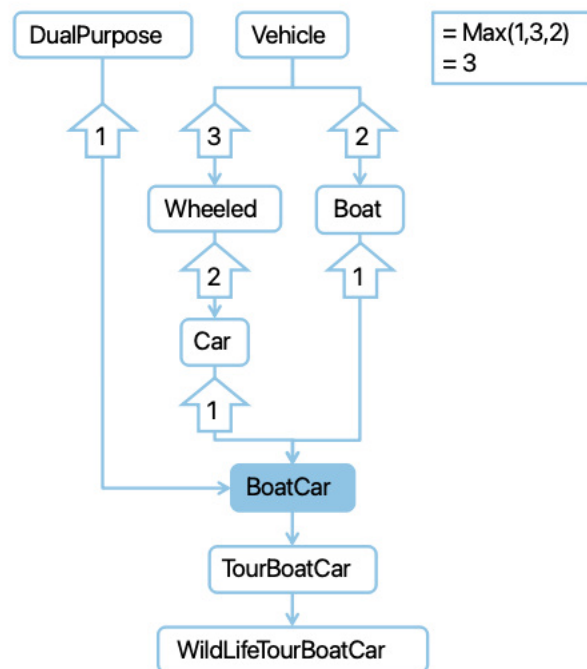
Languages: C#, C++, Java, Pascal, Python, Web

Targets: Classes

Maximum depth of class in inheritance tree.

The depth of a class within the inheritance hierarchy is the maximum number of nodes from the class node to the root of the inheritance tree. The root node has a DIT of 0. The deeper within the hierarchy, the more methods the class can inherit, increasing its complexity

Also known as Chidamber & Kemerer - Depth of Inheritance Tree (DIT).



Targets By Language:

- **C#:** Class, Struct
- **C++:** Class, Struct, Union
- **Java:** Class, Interface
- **Pascal:** Class, Interface
- **Python:** Class
- **Web:** PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Object

Orientated" with the "Show inheritance metrics" option.

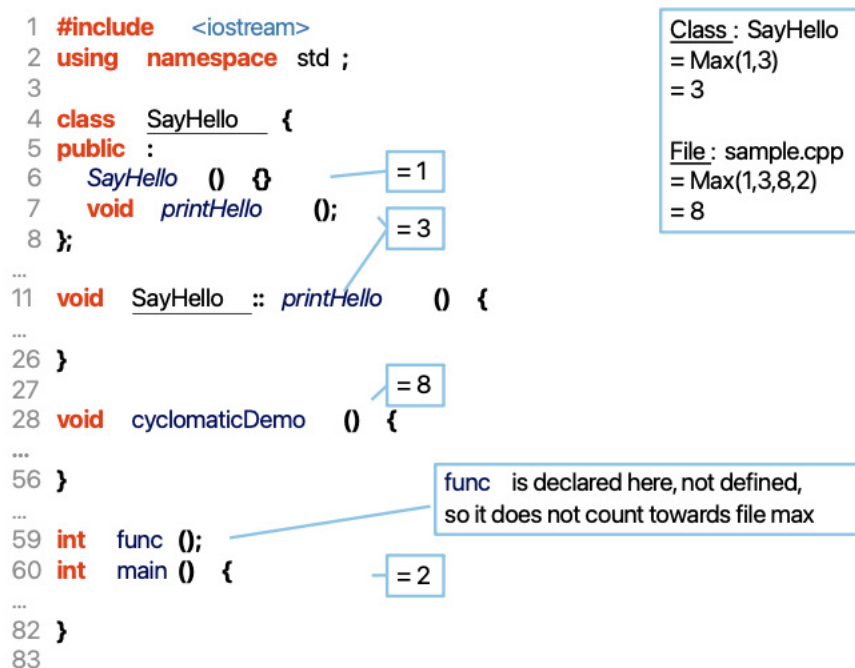
Max Modified Cyclomatic Complexity

API ID: MaxCyclomaticModified

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Architectures, Classes, Files, Modules, Packages, Project

Maximum modified cyclomatic complexity of nested functions or methods.



```

1  #include <iostream>
2  using namespace std ;
3
4  class SayHello {
5  public :
6      SayHello () {}
7      void printHello () ;
8  };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {
...
56 }
...
59 int func () ;
60 int main () {
...
82 }
83

```

Annotations:

- Class : SayHello**
= Max(1,3)
= 3
- File : sample.cpp**
= Max(1,3,8,2)
= 8
- func** is declared here, not defined, so it does not count towards file max

Complexity Values:

- Line 6: **= 1**
- Line 7: **= 3**
- Line 28: **= 8**
- Line 60: **= 2**

Targets By Language:

- **Ada:** Project, File, Package
- **Basic:** Project, File, Module, Class, Struct
- **C#:** Project, File, Class, Struct
- **C++:** Project, File, Class, Struct, Union
- **Fortran:** Project, File
- **Java:** Project, File, Class, Interface
- **Jovial:** Project, File
- **Pascal:** Project, File, Class, Interface, Compunit
- **Python:** Project, File, Class
- **Rust:** Project, File
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- The cyclomatic metric used is configured from "Project Configuration/Metrics/Complexity". This metric results from having "Count each decision in a multi-way-decision structure" off and having "Count logical conjunctions (OR) and logical and (AND)" off. Checking "Show all available cyclomatic complexity metric variants" will also enable the metric.

Max Nesting

API ID: MaxNesting

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Architectures, Classes, Files, Functions, Modules, Packages, Project, Subprograms

Maximum nesting level of control constructs (if, while, for, switch, etc.) in the function.

```

Nesting
0 28 void cyclomaticDemo () {
0 29     bool a = true , b = true , c = true ;
0 30
1 31     if ( a || ( b && c ) ) {
2 32         while ( a ? b : c ) {
3 33             for ( int i = 0; i < 10; i++ ) {
4 34                 switch ( i ) {
4 35                     case 1:
4 36                     case 2:
4 37                         cout << <<endl ;
4 38                         break ;
4 39                     case 5:
4 40                         break ;
4 41                     default :
4 42                         cout << <<endl ;
4 43                 }
3 44             }
2 45         }
1 46     } else {
1 47         try {
2 48             do {
2 49                 cout << a << b << c << endl ;
2 50             } while ( a );
1 51         }
1 52         catch (...){}
1 53     }
1 54 }
1 55 }
0 56 }
  
```

= 4

Targets By Language:

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Project, File, Method, Module, Class, Struct
- **C#:** Project, File, Class, Struct, Method
- **C++:** Project, File, Class, Struct, Union, Function

- **Fortran:** Project, File, Module, Block Data, Function, Program, Subroutine
- **Java:** Project, File, Class, Interface, Method
- **Jovial:** Project, File, Subroutine
- **Pascal:** Project, File, Class, Interface, Compunit, Function, Procedure
- **Python:** Project, File, Class, Function
- **Rust:** Project, File, Function
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- This metric can be disabled from "Project Configuration/Metrics/Complexity" with the "Maximum Nesting" option.

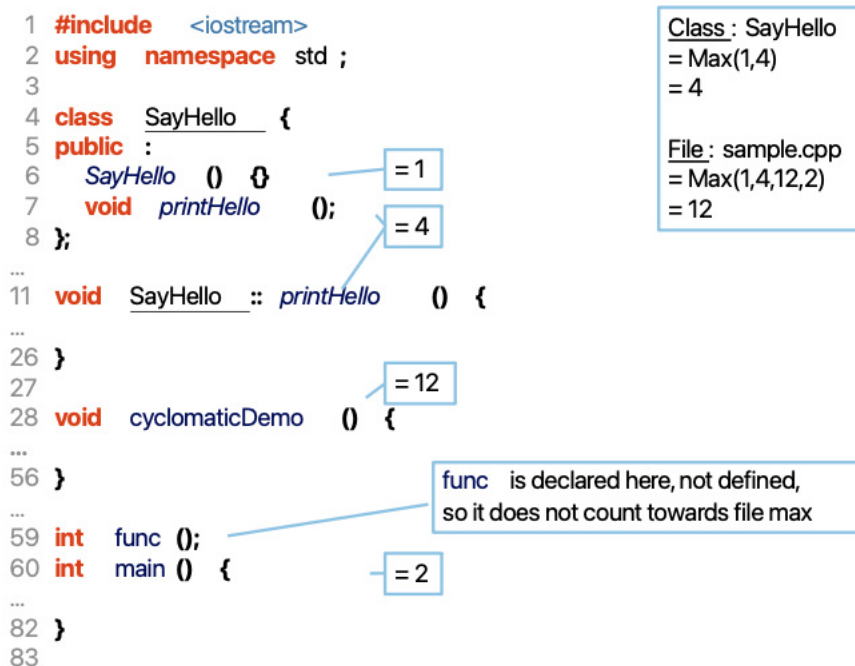
Max Strict Cyclomatic Complexity

API ID: MaxCyclomaticStrict

Languages: Ada, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Architectures, Classes, Files, Modules, Packages, Project

Maximum strict cyclomatic complexity of nested functions or methods.



```

1 #include <iostream>
2 using namespace std ;
3
4 class SayHello {
5 public :
6     SayHello () {}
7     void printHello () ;
8 };
9
10
11 void SayHello :: printHello () {
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26 }
27
28 void cyclomaticDemo () {
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56 }
57
58
59 int func () ;
60 int main () {
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82 }
83

```

Class : SayHello
 = Max(1,4)
 = 4

File : sample.cpp
 = Max(1,4,12,2)
 = 12

func is declared here, not defined,
 so it does not count towards file max

Targets By Language:

- **Ada:** Project, File, Package
- **C#:** Project, File, Class, Struct
- **C++:** Project, File, Class, Struct, Union
- **Fortran:** Project, File
- **Java:** Project, File, Class, Interface
- **Jovial:** Project, File
- **Pascal:** Project, File, Class, Interface, Compunit
- **Python:** Project, File, Class
- **Rust:** Project, File
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- The cyclomatic metric used is configured from "Project Configuration/Metrics/Complexity". This metric results from having "Count each decision in a multi-way-decision structure" on and having "Count logical conjunctions (OR) and logical and (AND)" on. Checking "Show all available cyclomatic complexity metric variants" will also enable the metric.

Max Strict Modified Cyclomatic Complexity

API ID: MaxCyclomaticStrictModified

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Architectures, Classes, Files, Modules, Packages, Project

Maximum strict modified cyclomatic complexity of nested functions or methods.

```

1  #include <iostream>
2  using namespace std ;
3
4  class SayHello {
5  public :
6      SayHello () {}
7      void printHello () ;
8  };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {
...
56 }
...
59 int func () ;
60 int main () {
...
82 }
83

```

Annotations for Cyclomatic Complexity:

- Line 6: `SayHello () {}` → = 1
- Line 7: `void printHello () ;` → = 3
- Line 28: `void cyclomaticDemo () {` → = 10
- Line 59: `int func () ;` → = 2

Class : SayHello
 = Max(1,3)
 = 3

File : sample.cpp
 = Max(1,3,10,2)
 = 10

func is declared here, not defined,
 so it does not count towards file max

Targets By Language:

- **Ada:** Project, File, Package
- **Basic:** Project, File, Module, Class, Struct
- **C#:** Project, File, Class, Struct
- **C++:** Project, File, Class, Struct, Union
- **Fortran:** Project, File
- **Java:** Project, File, Class, Interface
- **Jovial:** Project, File
- **Pascal:** Project, File, Class, Interface, Compunit
- **Python:** Project, File, Class
- **Rust:** Project, File
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- The cyclomatic metric used is configured from "Project Configuration/Metrics/Complexity". This metric results from having "Count each decision in a multi-way-decision structure" off and having "Count logical conjunctions (OR) and logical and (AND)" on. Checking "Show all available cyclomatic complexity metric variants" will also enable the metric.

Max Strict Modified Essential Complexity

API ID: MaxEssentialStrictModified

Languages: Ada

Targets: Architectures, Files, Packages, Project

Maximum strict modified essential complexity of all nested functions or methods.

Targets By Language:

- **Ada:** Project, File, Package

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Complexity" with the "Essential" option.

Methods

API ID: CountDeclMethod

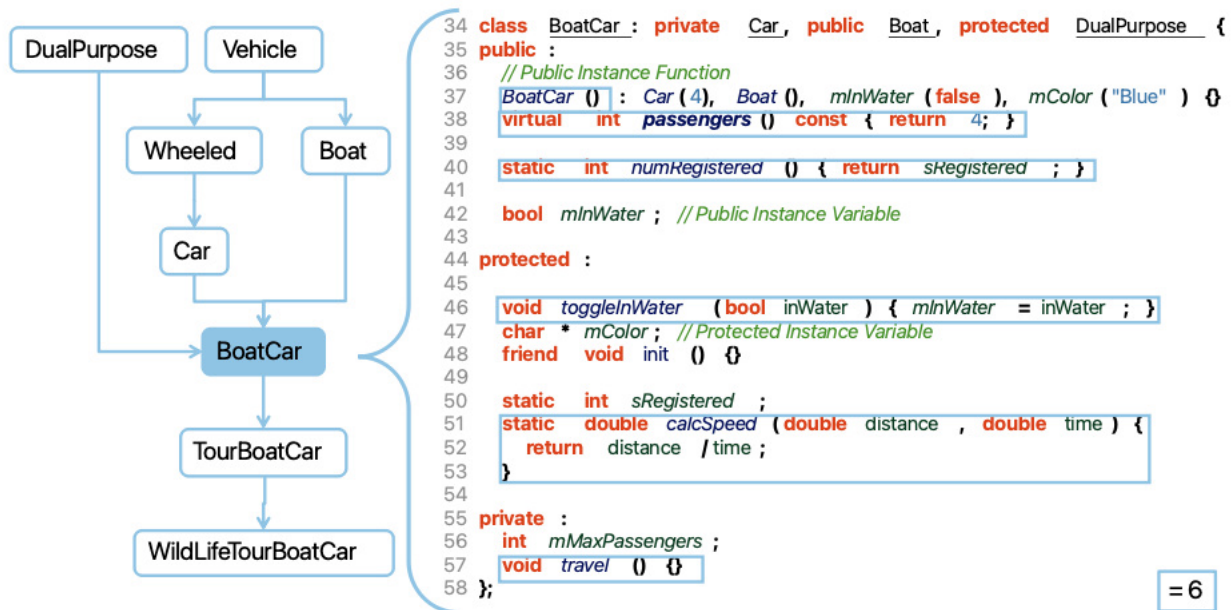
Also Known As: Weighted Methods per Class (WMC)

Languages: Basic, C#, C++, Java, Pascal, Python, Web

Targets: Architectures, Classes, Files, Modules, Project

Number of local (not inherited) methods.

Also known as Chidamber & Kemerer - Weighted Methods per Class (WMC).



= 6

Targets By Language:

- **Basic:** Project, Module, Class, Struct
- **C#:** Project, Class, Struct
- **C++:** Project, Class, Struct, Union
- **Java:** Project, File, Class, Interface
- **Pascal:** Project, Class, Interface
- **Python:** Project, Class
- **Web:** Project, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

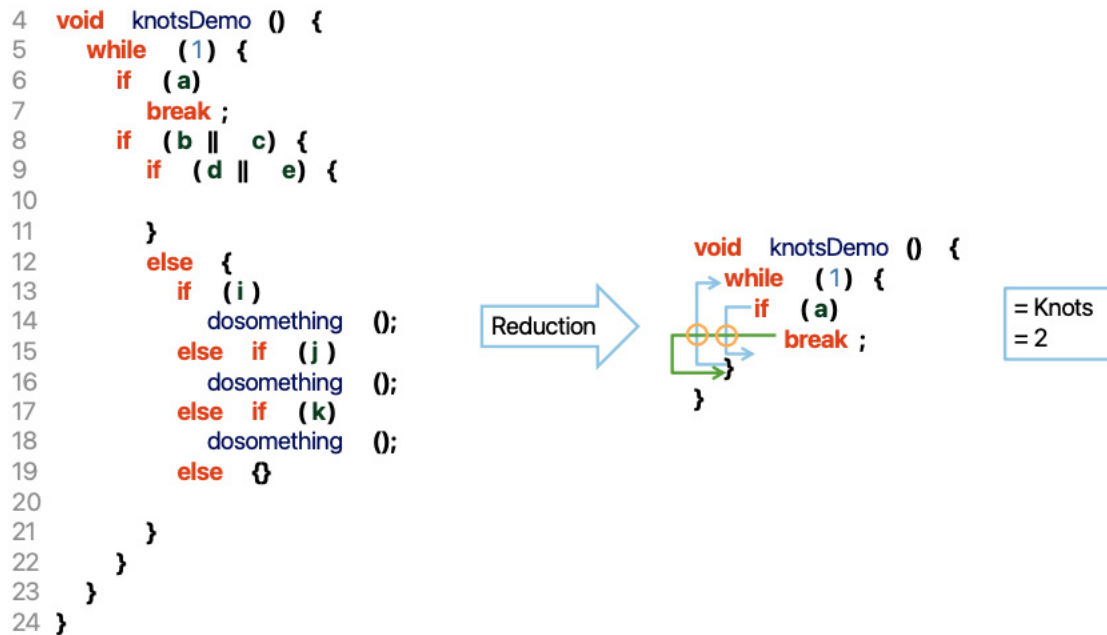
Min Essential Knots

API ID: MinEssentialKnots

Languages: Ada, C#, C++, Java, Rust

Targets: Functions, Packages, Subprograms

Minimum Knots after structured programming constructs have been removed.



Targets By Language:

- **Ada:** Type, Entry, Function, Package, Procedure, Protected, Task
- **C#:** Method
- **C++:** Function
- **Java:** Method
- **Rust:** Function

Configuration:

- This metric can be enabled from "Project Configuration/Metrics/Complexity" with the "Knots" option.

Modified Cyclomatic Complexity

API ID: CyclomaticModified

Also Known As: Modified Cyclomatic Complexity (CC3)

Languages: Ada, Assembly, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, VHDL, Web

Targets: Files, Functions, Modules, Packages, Subprograms

Modified McCabe Cyclomatic complexity.

The Cyclomatic Complexity except that each decision in a multi-decision structure (switch in C/Java, Case in Ada, computed Goto and arithmetic if in FORTRAN)

statement is not counted and instead the entire multi-way decision structure counts as 1.

Also known as McCabe - Modified Cyclomatic Complexity (CC3).

SWITCH	CATCH	DO	FOR	IF	?	WHILE	AND	OR	
✓	✓		✓	✓	✓	✓	✓	✓	28 void cyclomaticDemo () {
									29 bool a = true, b = true, c = true;
									30
									31 if (a (b && c)) {
									32 while (a ? b : c) {
									33 for (int i = 0; i < 10; i++) {
									34 switch (i) {
									35 case 1:
									36 case 2:
									37 cout << i << endl;
									38 break;
									39 case 5:
									40 break;
									41 default :
									42 cout << i << endl;
									43 }
									44 }
									45 }
									46 } else {
									47 try {
									48 do {
									49 cout << a << b << c << endl;
									50 } while (a);
									51 }
									52 catch (...){
									53 }
									54 }
									55 }
									56 }
Modified	Not Modified	Always					Strict		
1	3	1	1	1	1	1	1	1	

= decision points + 1
 = 7 + 1
 = 8

Targets By Language:

- **Ada:** Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Method
- **C#:** Method
- **C++:** Function
- **Fortran:** Module, Block Data, Function, Program, Subroutine
- **Java:** Method
- **Jovial:** Subroutine
- **Pascal:** Compunit, Function, Procedure
- **Python:** File, Function
- **Rust:** Function
- **VHDL:** Procedure, Function, Process
- **Web:** File

Configuration:

- The cyclomatic metric used is configured from "Project Configuration/ Metrics/Complexity". This metric results from having "Count each decision in a multi-way-decision structure" off and having "Count logical conjunctions (OR) and logical and (AND)" off. Checking "Show all available cyclomatic complexity metric variants" will also enable the metric.

Modules

API ID: CountDeclModule

Languages: Fortran, Jovial, Pascal

Targets: Architectures, Files, Functions, Modules, Project, Subprograms

Number of modules.

Targets By Language:

- **Fortran:** Project, File, Module, Block Data, Function, Program, Subroutine
- **Jovial:** Project, File
- **Pascal:** Project, File

Configuration:

- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated file metrics" option.

New Lines

API ID: CountLineNew

Languages: Ada, Assembly, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, VHDL, Web

Targets: Classes, Files, Functions, Modules, Packages, Subprograms

Number of new lines

Outputs

API ID: CountOutput

Also Known As: FANOUT (Infomational fan-out)

Languages: C#, C++, Fortran, Java

Targets: Functions, Subprograms

Number of called subprograms plus global variables set.

The number of outputs that are SET. These can be parameters or global variables. So Functions calls + Parameters set/modify + Global Variables set/modify. A non-void return value adds one to the count. Recursive function calls, local variables that are not class static variables, and parameters that are pass by value are not included. Of the two general approaches to calculating FANOUT (informational versus

structural) ours is the informational approach.

Also known as FANOUT (Infomational fan-out).

```

3  int  in  = 1;
4  int  out = 1;
5
6  int  inOutFunc (int  in1 , int  in2 , int  *inout1 , int  &inout2 , int  * out1 , int  & out2 ) {
7      out = in  + in1  + in2  + *inout1  + inout2 ;
8
9      *inout1  = in1 ;
10     inout2   = in2 ;
11
12     *out1    = in1 ;
13     out2     = in2 ;
14
15     in1      = somefunc ();
16     in2      = 2;
17
18     int  randomint  = 3;
19     in1   = randomint ;
20
21     return 4;
22 }

```

= functions called + parameters set +
globals set + non -void return type
= 1+4+1+1
= 7

Entity	Counts?	Comment
in	No	Not set
out	Yes	Set line 7
in1	No	Pass by value does not count
in2	No	Pass by value does not count
inout1	Yes	Set line 9
inout2	Yes	Set line 10
out1	Yes	Set line 12
out2	Yes	Set line 13
randomint	No	Not a parameter, global, or class static variable
somefunc	Yes	Non-recursive function call, line 15

Targets By Language:

- **C#:** Method
- **C++:** Function
- **Fortran:** Function, Program, Subroutine
- **Java:** Method

Paths

API ID: CountPath

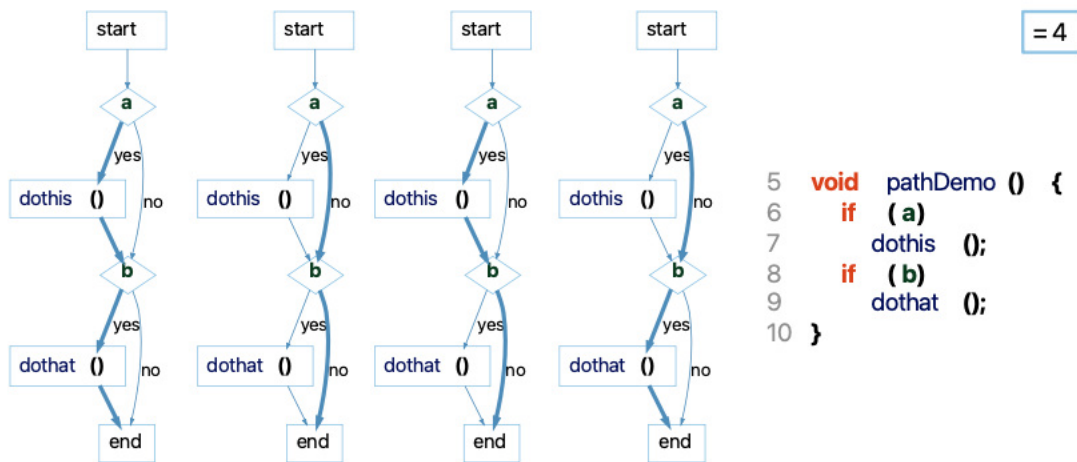
Also Known As: NPATH

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Files, Functions, Modules, Packages, Subprograms

Number of unique paths through a body of code, not counting abnormal exits or gotos.

Also known as NPATH.



Targets By Language:

- **Ada:** Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Method
- **C#:** Method
- **C++:** Function
- **Fortran:** Module, Block Data, Function, Program, Subroutine
- **Java:** Method
- **Jovial:** Subroutine
- **Pascal:** Compunit, Function, Procedure
- **Python:** File, Function
- **Rust:** File, Function
- **Web:** File

Configuration:

- This metric can be enabled from "Project Configuration/Metrics/Complexity" with the "Paths" option.

Paths Log(x)

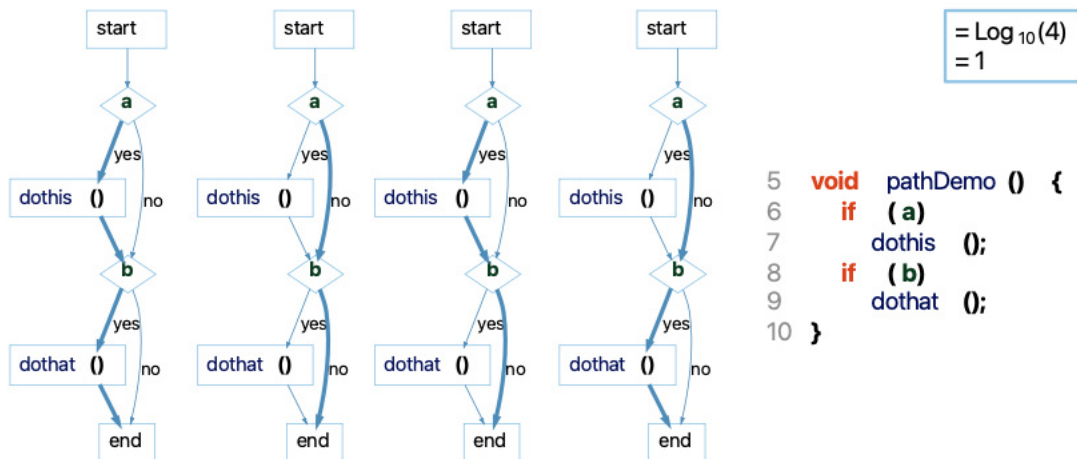
API ID: CountPathLog

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Files, Functions, Modules, Packages, Subprograms

The base 10 logarithm $\text{Log}(x)$ of the number of unique paths through a body of code,

not counting abnormal exits or gotos, truncated to an integer value.



Targets By Language:

- **Ada:** Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Method
- **C#:** Method
- **C++:** Function
- **Fortran:** Module, Block Data, Function, Program, Subroutine
- **Java:** Method
- **Jovial:** Subroutine
- **Pascal:** Compunit, Function, Procedure
- **Python:** File, Function
- **Rust:** File, Function
- **Web:** File

Configuration:

- This metric can be enabled from "Project Configuration/Metrics/Complexity" with the "Paths" option.

Percent Changed

API ID: PercentChanged

Languages: Ada, Assembly, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, VHDL, Web

Targets: Classes, Files, Functions, Modules, Packages, Subprograms

$(\text{Added} + \text{Removed} + 2 * \text{Changed}) / (\text{CurrentLines} + \text{PreviousLines})$

Percent Lack Of Cohesion

API ID: PercentLackOfCohesion

Also Known As: LCOM2, Lack of Cohesion in Methods (LCOM/LOCM)

Languages: Ada, Basic, C#, C++, Java, Pascal

Targets: Classes, Packages

100% minus the average cohesion for package entities.

100% minus average cohesion for class data members. Calculates what percentage of class methods use a given class instance variable. To calculate, average percentages for all of that class's instance variables and subtract from 100%. A lower percentage means higher cohesion between class data and methods.

Also known as Chidamber & Kemerer - Lack of Cohesion in Methods (LCOM/LOCM); LCOM2.

```

1 class CohesionClass {
2 public :
3     void func1 () {
4         for (int i = 0; i < mVar1; i++) {
5             mVar2 = nullptr ;
6         }
7         mVar1 = 3;
8     }
9
10    void func2 () {
11        mVar1 = 4;
12    }
13
14    static void addObj () {
15        sNumObjs++;
16    }
17 protected :
18
19    void func3 () {
20        mVar2 = "blue" ;
21    }
22 private :
23
24    void func4 () {
25    }
26
27
28    int mVar1;
29    char * mVar2;
30    static int sNumObjs;
31 };

```

	mVar1	mVar2	sNumObjs
func1 ()	✓	✓	
func2 ()	✓		
addObj ()			✓
func3 ()		✓	
func4 ()			

# Functions Using Variable:	2	2	1
Divided By Total Functions (5):	0.4	0.4	0.2
Averaged Together:	0.3333		
Subtract from 1:	0.6667		
To Percent:	67%		

Targets By Language:

- **Ada:** Package
- **Basic:** Class, Struct

- **C#:** Class, Struct
- **C++:** Class, Struct, Union
- **Java:** Class, Interface
- **Pascal:** Class, Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show coupling and cohesion metrics" option.

Percent Lack Of Cohesion Modified

API ID: PercentLackOfCohesionModified

Languages: Basic, C#, Java, Pascal

Targets: Classes

100% minus the average cohesion for class data members, modified for accessor methods.

Same as PercentLackOfCohesion but does not penalize the use of accessor methods within a class to set/read variables.

Targets By Language:

- **Basic:** Class, Struct
- **C#:** Class, Struct
- **Java:** Class, Interface
- **Pascal:** Class, Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show coupling and cohesion metrics" option.

Preprocessor Lines

API ID: CountLinePreprocessor

Languages: Ada, C#, C++

Targets: Architectures, Classes, Files, Functions, Packages, Project, Subprograms

Number of preprocessor lines.

Code	Comment	Preprocessor	Declarative	Executable	Inactive	
✓		✓				11 void SayHello :: <i>printHello</i> () {
✓			✓			12 switch (i) {
✓				✓		13 case 0:
✓				✓		14 cout << "Hello World" << endl ;
✓				✓		15 case 1:
✓				✓		16 cout << "HELLO WORLD!" << endl ;
✓	✓			✓		17 default : <i>// a comment here</i>
✓			✓	✓		18 for (int m = 0; m < j; m++);
✓				✓		19 cout << "hello world" << endl ;
✓						20 }
	✓					21 #ifdef A_VERY_NICE_VARIABLE
					✓	22
✓	✓			✓		23 cout << "Inactive Line" << endl ; <i>// Inactive</i>
	✓					24 #endif
						25
✓						26 }

= Preprocessor
= 2

Targets By Language:

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **C#:** Project, File, Class, Method
- **C++:** Project, File, Class, Struct, Union, Function

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show line count metrics" option.
- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show default summary metrics" option.

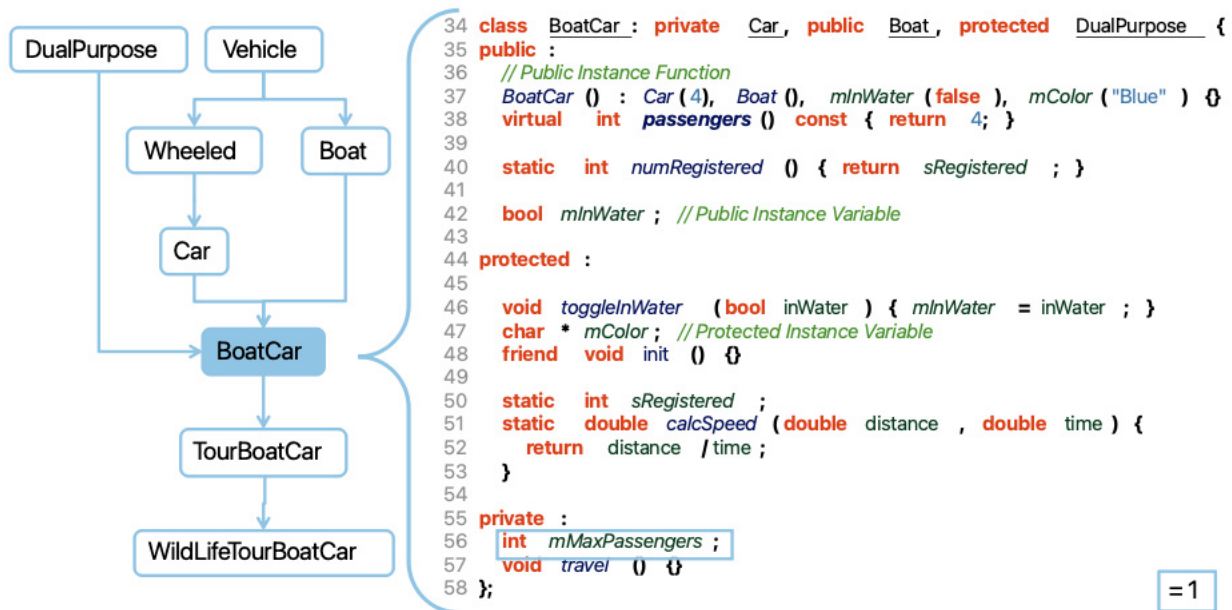
Private Instance Variables

API ID: CountDeclInstanceVariablePrivate

Languages: C#, C++, Web

Targets: Architectures, Classes, Project

Number of private instance variables.



Targets By Language:

- **C#:** Project, Class, Struct
- **C++:** Project, Class, Struct, Union
- **Web:** Project, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Private Methods

API ID: CountDeclMethodPrivate

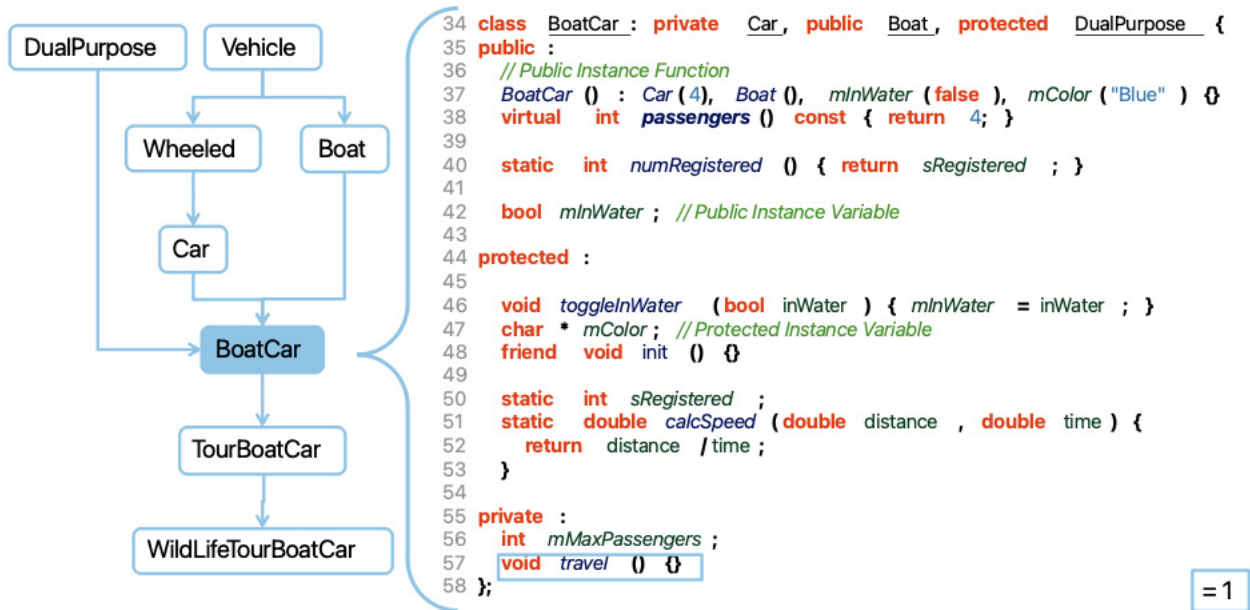
Also Known As: Number Private Methods (NPRM)

Languages: Basic, C#, C++, Java, Pascal, Web

Targets: Architectures, Classes, Files, Modules, Project

Number of local (not inherited) private methods.

Also known as Number Private Methods (NPRM).



Targets By Language:

- **Basic:** Project, Module, Class, Struct
- **C#:** Project, Class, Struct
- **C++:** Project, Class, Struct, Union
- **Java:** Project, File, Class, Interface
- **Pascal:** Project, Class, Interface
- **Web:** Project, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Program Units

API ID: CountDeclProgUnit

Languages: Fortran

Targets: Architectures, Files, Project

Number of non-nested modules, block data units, and subprograms.

Targets By Language:

- **Fortran:** Project, File

Configuration:

- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated file metrics" option.

Properties

API ID: CountDeclProperty

Languages: C#, Pascal

Targets: Architectures, Classes, Project

Number of properties.

Targets By Language:

- **C#:** Project, Class, Struct
- **Pascal:** Project, Class, Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

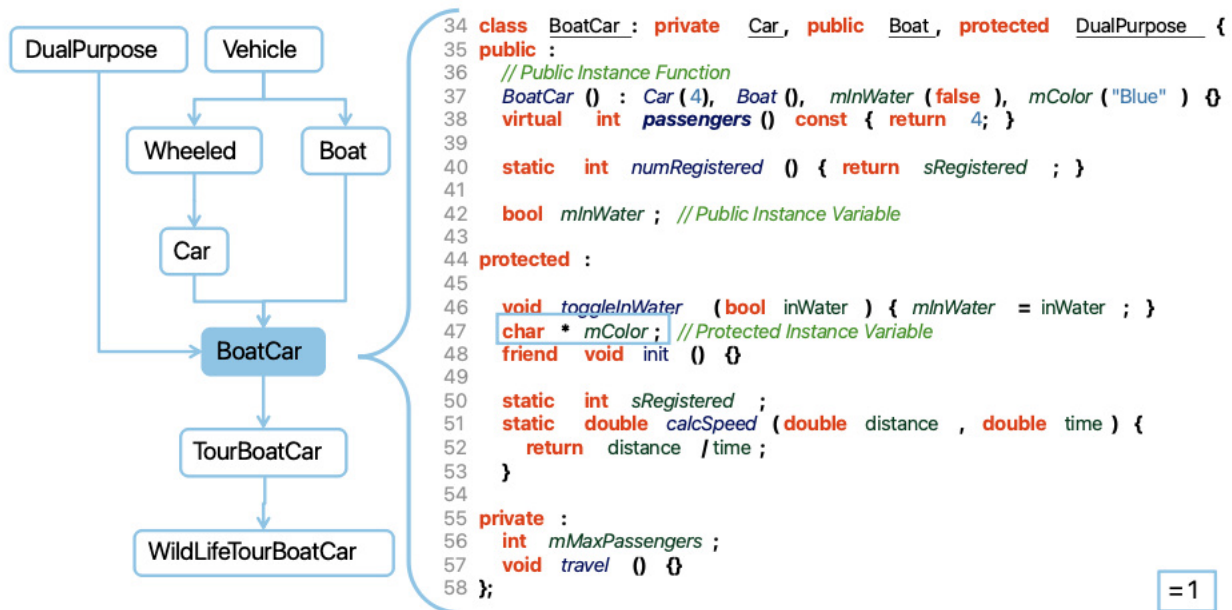
Protected Instance Variables

API ID: CountDeclInstanceVariableProtected

Languages: C#, C++, Web

Targets: Architectures, Classes, Project

Number of protected instance variables.



Targets By Language:

- **C#:** Project, Class, Struct
- **C++:** Project, Class, Struct, Union
- **Web:** Project, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Protected Internal Instance Variables

API ID: CountDeclInstanceVariableProtectedInternal

Languages: C#

Targets: Architectures, Classes, Project

Number of protected internal instance variables.

Targets By Language:

- **C#:** Project, Class, Struct

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Protected Internal Methods

API ID: CountDeclMethodProtectedInternal

Languages: C#

Targets: Architectures, Classes, Project

Number of local protected internal methods.

Targets By Language:

- **C#:** Project, Class, Struct

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

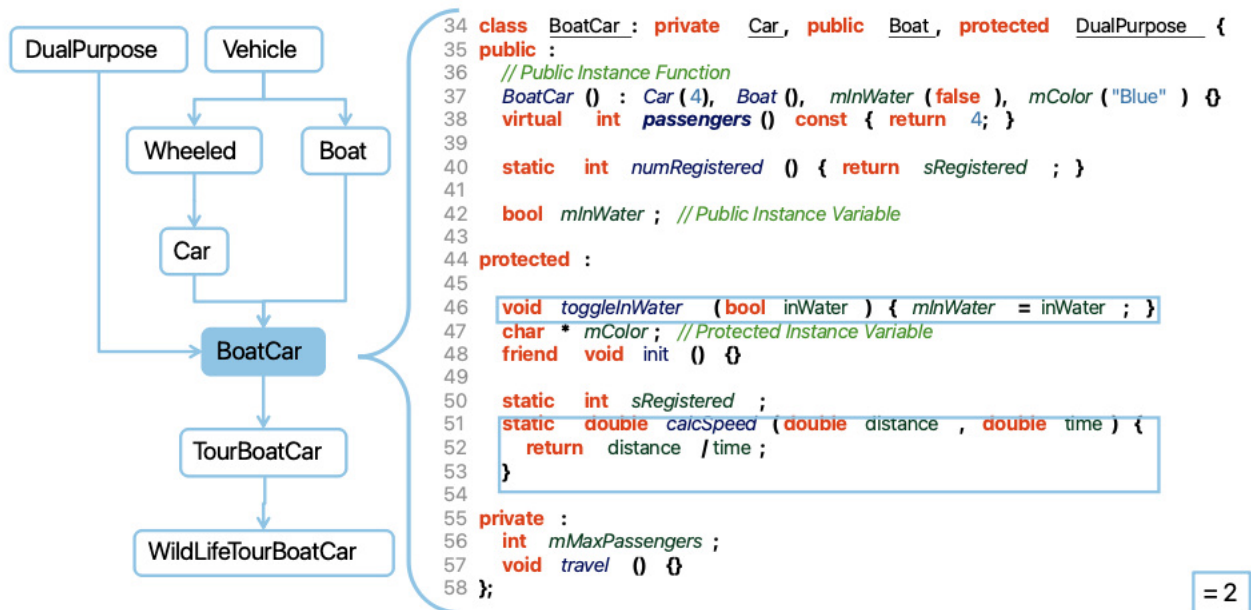
Protected Methods

API ID: CountDeclMethodProtected

Languages: Basic, C#, C++, Java, Pascal, Web

Targets: Architectures, Classes, Files, Modules, Project

Number of local protected methods.



Targets By Language:

- **Basic:** Project, Module, Class, Struct
- **C#:** Project, Class, Struct
- **C++:** Project, Class, Struct, Union
- **Java:** Project, File, Class, Interface
- **Pascal:** Project, Class, Interface
- **Web:** Project, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

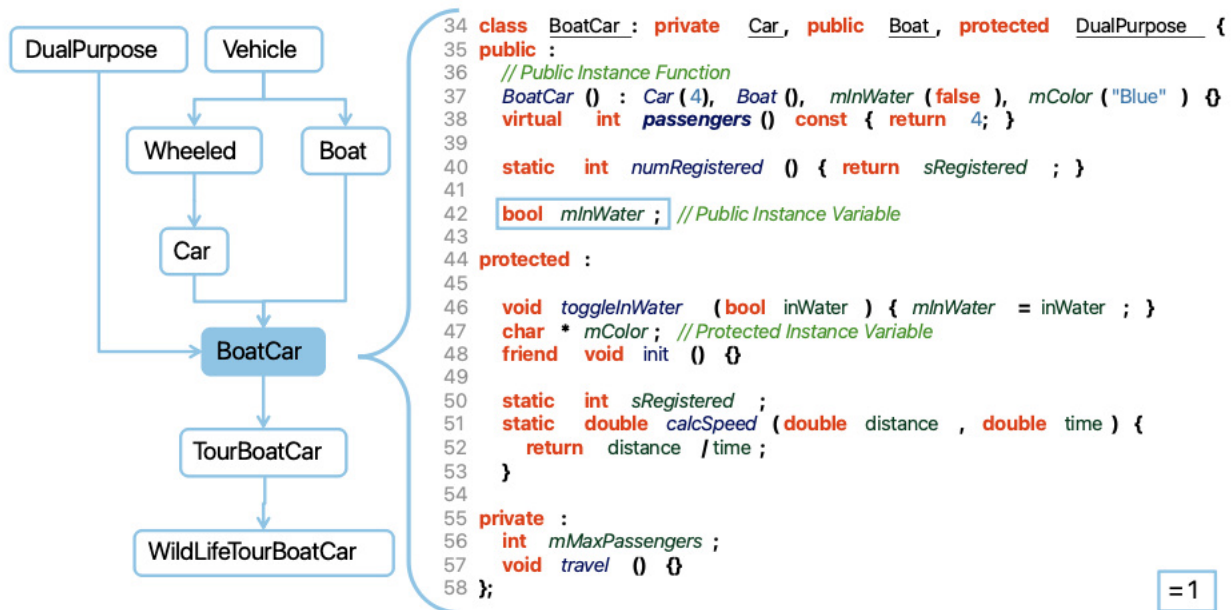
Public Instance Variables

API ID: CountDeclInstanceVariablePublic

Languages: C#, C++, Web

Targets: Architectures, Classes, Project

Number of public instance variables.



= 1

Targets By Language:

- **C#:** Project, Class, Struct
- **C++:** Project, Class, Struct, Union
- **Web:** Project, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Public Methods

API ID: CountDeclMethodPublic

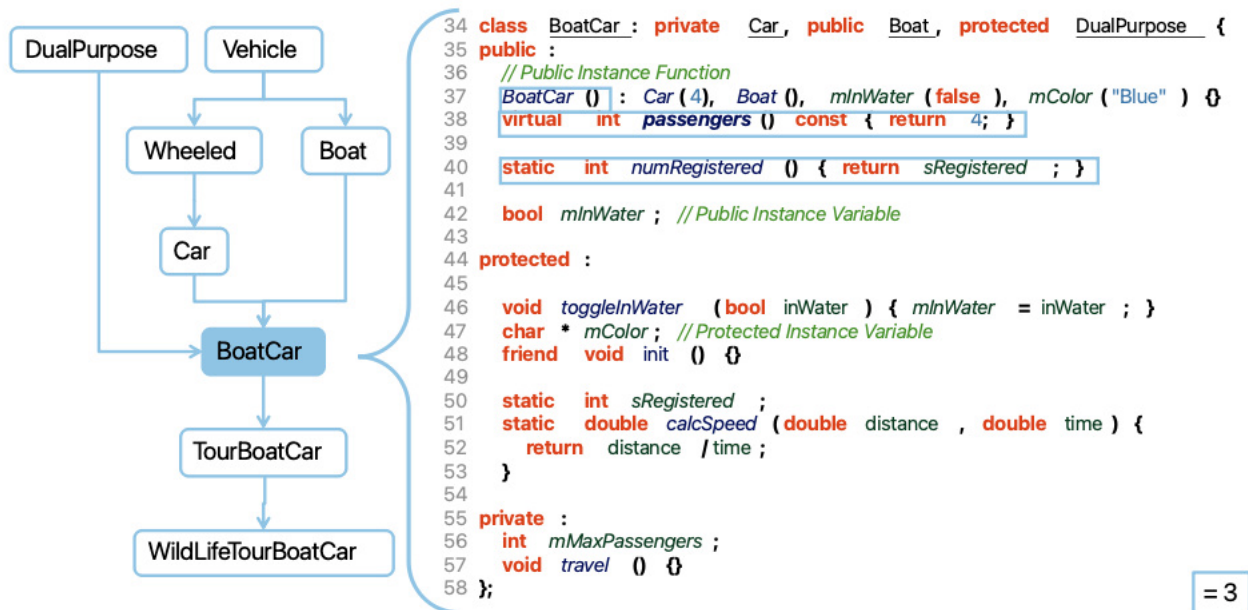
Also Known As: Number of Public Methods (PM, NPM)

Languages: Basic, C#, C++, Java, Pascal, Web

Targets: Architectures, Classes, Files, Modules, Project

Number of local (not inherited) public methods.

Also known as Lorenz & Kidd - Number of Public Methods (PM, NPM).



Targets By Language:

- **Basic:** Project, Module, Class, Struct
- **C#:** Project, Class, Struct
- **C++:** Project, Class, Struct, Union
- **Java:** Project, File, Class, Interface
- **Pascal:** Project, Class, Interface
- **Web:** Project, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Removed Lines

API ID: CountLineRemoved

Languages: Ada, Assembly, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, VHDL, Web

Targets: Classes, Files, Functions, Modules, Packages, Subprograms

Number of removed lines

Semicolons

API ID: CountSemicolon

Languages: Ada, C#, C++, Java

Targets: Architectures, Classes, Files, Functions, Packages, Project, Subprograms

Number of semicolons.

```

11 void SayHello :: printHello () {
12     switch (i) {
13         case 0:
14             cout << "Hello World" << endl ;
15         case 1:
16             cout << "HELLO WORLD!" << endl ;
17         default : //a comment here
18             for (int m = 0; m < j; m++)
19                 cout << "hello world" << endl ;
20     }
21     #ifdef A_VERY_NICE_VARIABLE
22         cout << "Inactive Line" << endl ; //Inactive
23     #endif
24
25
26 }

```

= 6

Targets By Language:

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **C#:** Project, File, Class, Method
- **C++:** Project, File, Function
- **Java:** Project, File, Class, Interface, Method

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show statement count metrics" option.

Statements

API ID: CountStmt

Languages: Ada, Assembly, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, VHDL, Web

Targets: Architectures, Classes, Files, Functions, Modules, Packages, Project, Subprograms

Number of statements.

		Executive	Declarative	Empty	Total
Statement is declarative, but only counts at file scope	60 int main () {	0	1	0	1
	61 for (int i = 0;	1	0	0	1
	62 i < 10;	1	0	0	1
	63 i ++)	0	0	1	1
	64 ;				
	65				
	66 int j = func 0;	1 (0)	1	0	1
	67 int k = 0;	0	1	0	1
Initializations with function calls are both declarative and executable in strict	68 int l = 1;	0	1	0	1
	69				
	70 int m	0	1	0	1
	71 = func 0;	1 (0)	0	0	0
	72 int n;	0	1	0	1
	73 n = 1;	1	0	0	1
	74				
Inactive code is not counted	75 #ifdef A_VERY_NICE_VARIABLE				
	76 int j = 0;				
	77 cout << j << endl ;				
	78				
	79 #endif				
	80				
	81 return 0;	1	0	0	1
	82 }	0	0	0	0
		6 (4)	6	1	11

Fuzzy values that differ from strict values are shown in parentheses.

Targets By Language:

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Project, File, Method, Module, Class
- **C#:** Project, File, Class, Method
- **C++:** Project, File, Class, Struct, Union, Function
- **Fortran:** Project, File, Block Data, Function, Program, Subroutine
- **Java:** Project, File, Class, Interface, Method
- **Jovial:** Project, File, Module, Subroutine
- **Pascal:** Project, File, Class, Interface, Compunit, Function, Procedure
- **Python:** Project, File, Class, Function
- **Rust:** Project, File, Function
- **VHDL:** Project, File
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- This metric can be disabled from "Project Configuration/Metrics/Lines and Statements" with the "Show statement count metrics" option.

Strict Cyclomatic Complexity

API ID: CyclomaticStrict

Also Known As: Strict Cyclomatic Complexity (CC2)

Languages: Ada, Assembly, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, VHDL, Web

Targets: Files, Functions, Modules, Packages, Subprograms

Strict McCabe Cyclomatic Complexity.

The Cyclomatic Complexity with logical conjunction and logical and in conditional expressions also adding 1 to the complexity for each of their occurrences. i.e. The statement 'if (a && b || c)' would have a cyclomatic complexity of 1 but a strict cyclomatic complexity of 3

Also known as McCabe - Strict Cyclomatic Complexity (CC2).

SWITCH	CASE	CATCH	DO	FOR	IF	?	WHILE	AND	OR	
					✓		✓	✓		28 void cyclomaticDemo () {
										29 bool a = true , b = true , c = true ;
										30
					✓					31 if (a (b && c)) {
						✓				32 while (a ? b : c) {
							✓			33 for (int i = 0; i < 10; i++) {
										34 switch (i) {
										35 case 1:
										36 case 2:
										37 cout << i << endl ;
										38 break ;
										39 case 5:
										40 break ;
										41 default :
										42 cout << i << endl ;
										43 }
										44 }
										45 }
										46 } else {
										47 try {
										48 do {
										49 cout << a << b << c << endl ;
										50 } while (a);
										51 }
										52 catch (...){
										53 }
										54 }
										55 }
										56 }

Modified
Not Modified

Always
Strict

1 3 1 1 1 1 1 1 1 1

= decision points + 1
 = 11 + 1
 = 12

Targets By Language:

- **Ada:** Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Method
- **C#:** Method
- **C++:** Function
- **Fortran:** Module, Block Data, Function, Program, Subroutine
- **Java:** Method

- **Jovial:** Subroutine
- **Pascal:** Compunit, Function, Procedure
- **Python:** File, Function
- **Rust:** Function
- **VHDL:** Procedure, Function, Process
- **Web:** File

Configuration:

- The cyclomatic metric used is configured from "Project Configuration/Metrics/Complexity". This metric results from having "Count each decision in a multi-way-decision structure" on and having "Count logical conjunctions (OR) and logical and (AND)" on. Checking "Show all available cyclomatic complexity metric variants" will also enable the metric.

Strict Modified Cyclomatic Complexity

API ID: CyclomaticStrictModified

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Files, Functions, Modules, Packages, Subprograms

Cyclomatic Complexity with the following conditions.

- (1) Logical operators (AND, OR) in conditional expressions add one (1) to the complexity for each occurrence.
- (2) In multi-decision structures (switch in C/Java, Case in Ada, computed Goto, and arithmetic if in FORTRAN) the decision points are not counted, instead, the entire multi-way decision structure counts as 1

SWITCH	CATCH	DO	FOR	IF	?	WHILE	AND	OR	
✓	✓	✓	✓	✓	✓	✓	✓	✓	28 void cyclomaticDemo () {
									29 bool a = true , b = true , c = true ;
									30
									31 if (a (b && c)) {
									32 while (a ? b : c) {
									33 for (int i = 0; i < 10; i++) {
									34 switch (i) {
									35 case 1:
									36 case 2:
									37 cout << endl ;
									38 break ;
									39 case 5:
									40 break ;
									41 default :
									42 cout << endl ;
									43 }
									44 }
									45 }
									46 } else {
									47 try {
									48 do {
									49 cout << a << b << c << endl ;
									50 } while (a);
									51 }
									52 catch (...){
									53 }
									54 }
									55 }
									56 }
1	3	1	1	1	1	1	1	1	

= decision points + 1
= 9 + 1
= 10

Targets By Language:

- **Ada:** Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Method
- **C#:** Method
- **C++:** Function
- **Fortran:** Module, Block Data, Function, Program, Subroutine
- **Java:** Method
- **Jovial:** Subroutine
- **Pascal:** Compunit, Function, Procedure
- **Python:** File, Function
- **Rust:** Function
- **Web:** File

Configuration:

- The cyclomatic metric used is configured from "Project Configuration/ Metrics/Complexity". This metric results from having "Count each decision in a multi-way-decision structure" off and having "Count logical conjunctions (OR) and logical and (AND)" on. Checking "Show all available cyclomatic complexity metric variants" will also enable the metric.

Strict Modified Essential Complexity

API ID: EssentialStrictModified

Languages: Ada

Targets: Packages, Subprograms

The cyclomatic complexity with short circuit operators (and then/or else) as unstructured but only adds one for all structured paths through case statements after graph reduction.

Targets By Language:

- **Ada:** Type, Entry, Function, Package, Procedure, Protected, Task

Configuration:

- This metric can be enabled from "Project Configuration/Metrics/Complexity" with the "Essential" option.

Strict Private Methods

API ID: CountDeclMethodStrictPrivate

Languages: Pascal

Targets: Architectures, Classes, Project

Number of local strict private methods.

Targets By Language:

- **Pascal:** Project, Class, Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Strict Published Methods

API ID: CountDeclMethodStrictPublished

Languages: Pascal

Targets: Architectures, Classes, Project

Number of local strict published methods.

Targets By Language:

- **Pascal:** Project, Class, Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated class metrics" option. Enabling this option can be time consuming.
- This metric can be disabled from "Project Configuration/Metrics/Object Orientated" with the "Show method and variable count metrics" option.

Subprograms

API ID: CountDeclSubprogram

Languages: Ada, Basic, Fortran, Jovial, Pascal

Targets: Architectures, Files, Functions, Modules, Packages, Project, Subprograms

Number of subprograms.

Targets By Language:

- **Ada:** Project, File, Package
- **Basic:** Project, File
- **Fortran:** Project, File, Module, Block Data, Function, Program, Subroutine
- **Jovial:** Project, File, Module, Subroutine
- **Pascal:** Project, File, Compunit, Function, Procedure

Configuration:

- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated file metrics" option.
- For projects and architectures, this metric is enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show default summary metrics" option.

Sum Cyclomatic Complexity

API ID: SumCyclomatic

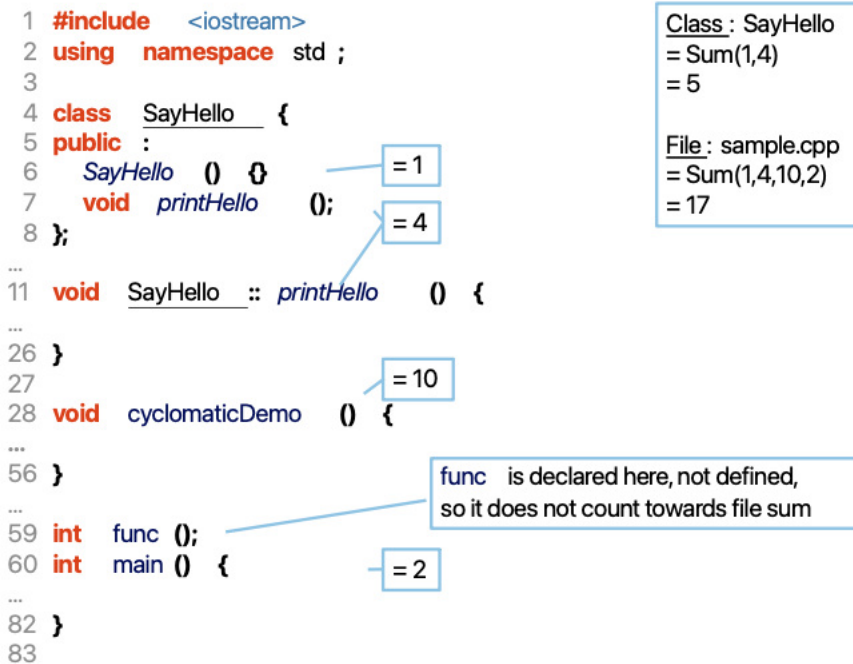
Also Known As: Weighted Methods per Class (WMC)

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Architectures, Classes, Files, Functions, Modules, Packages, Project, Subprograms

Sum of cyclomatic complexity of all nested functions or methods.

Also known as Chidamber & Kemerer - Weighted Methods per Class (WMC).



```

1  #include <iostream>
2  using namespace std ;
3
4  class SayHello {
5  public :
6      SayHello () {}           = 1
7      void printHello () ;     = 4
8  };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {     = 10
...
56 }
...
59 int func ();                = 2
60 int main () {
...
82 }
83

```

Class : SayHello
 = Sum(1,4)
 = 5

File : sample.cpp
 = Sum(1,4,10,2)
 = 17

func is declared here, not defined,
 so it does not count towards file sum

Targets By Language:

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Project, File, Method, Module, Class, Struct
- **C#:** Project, File, Class, Struct, Method
- **C++:** Project, File, Class, Struct, Union
- **Fortran:** Project, File, Module, Function, Program, Subroutine
- **Java:** Project, File, Class, Interface, Method
- **Jovial:** Project, File
- **Pascal:** Project, File, Class, Interface, Compunit, Function, Procedure
- **Python:** Project, File, Class
- **Rust:** Project, File
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- The cyclomatic metric used is configured from "Project Configuration/Metrics/Complexity". This metric results from having "Count each decision in a multi-way-decision structure" on and having "Count logical conjunctions (OR) and logical and (AND)" off. Checking "Show all available cyclomatic complexity metric variants" will also enable the metric.

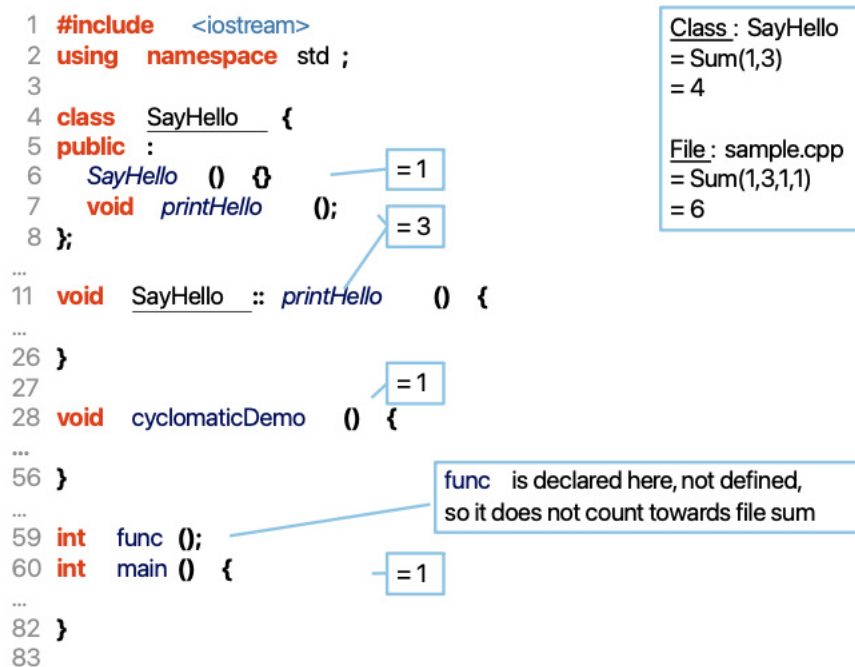
Sum Essential Complexity

API ID: SumEssential

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Architectures, Classes, Files, Functions, Modules, Packages, Project, Subprograms

Sum of essential complexity of all nested functions or methods.



```

1  #include <iostream>
2  using namespace std ;
3
4  class SayHello {
5  public :
6      SayHello () {}           = 1
7      void printHello () {}    = 3
8  };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {     = 1
...
56 }
...
59 int func ();
60 int main () {               = 1
...
82 }
83

```

Class : SayHello
= Sum(1,3)
= 4

File : sample.cpp
= Sum(1,3,1,1)
= 6

func is declared here, not defined,
so it does not count towards file sum

Targets By Language:

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Project, File, Method, Module, Class, Struct
- **C#:** Project, File, Class, Struct, Method
- **C++:** Project, File, Class, Struct, Union
- **Fortran:** Project, File, Module, Function, Program, Subroutine
- **Java:** Project, File, Class, Interface, Method
- **Jovial:** Project, File
- **Pascal:** Project, File, Class, Interface, Compunit, Function, Procedure
- **Python:** Project, File, Class
- **Rust:** Project, File
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Complexity" with the "Essential" option.

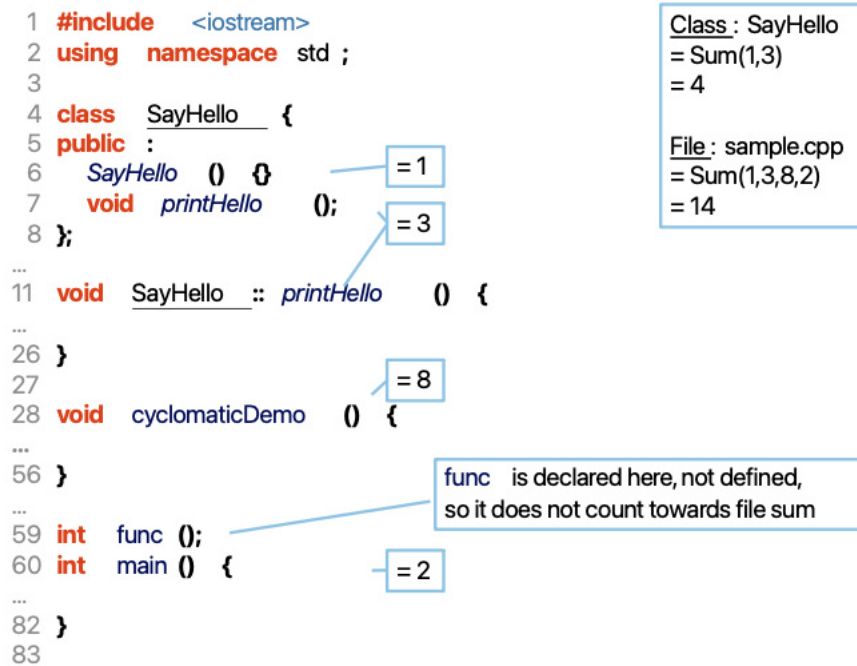
Sum Modified Cyclomatic Complexity

API ID: SumCyclomaticModified

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Architectures, Classes, Files, Functions, Modules, Packages, Project, Subprograms

Sum of modified cyclomatic complexity of all nested functions or methods.



```

1  #include <iostream>
2  using namespace std ;
3
4  class SayHello {
5  public :
6      SayHello () {}
7      void printHello () ;
8  };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {
...
56 }
...
59 int func ();
60 int main () {
...
82 }
83

```

Class : SayHello
= Sum(1,3)
= 4

File : sample.cpp
= Sum(1,3,8,2)
= 14

func is declared here, not defined,
so it does not count towards file sum

Targets By Language:

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Project, File, Method, Module, Class, Struct
- **C#:** Project, File, Class, Struct, Method
- **C++:** Project, File, Class, Struct, Union
- **Fortran:** Project, File, Module, Function, Program, Subroutine
- **Java:** Project, File, Class, Interface, Method
- **Jovial:** Project, File
- **Pascal:** Project, File, Class, Interface, Compunit, Function, Procedure

- **Python:** Project, File, Class
- **Rust:** Project, File
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- The cyclomatic metric used is configured from "Project Configuration/Metrics/Complexity". This metric results from having "Count each decision in a multi-way-decision structure" off and having "Count logical conjunctions (OR) and logical and (AND)" off. Checking "Show all available cyclomatic complexity metric variants" will also enable the metric.

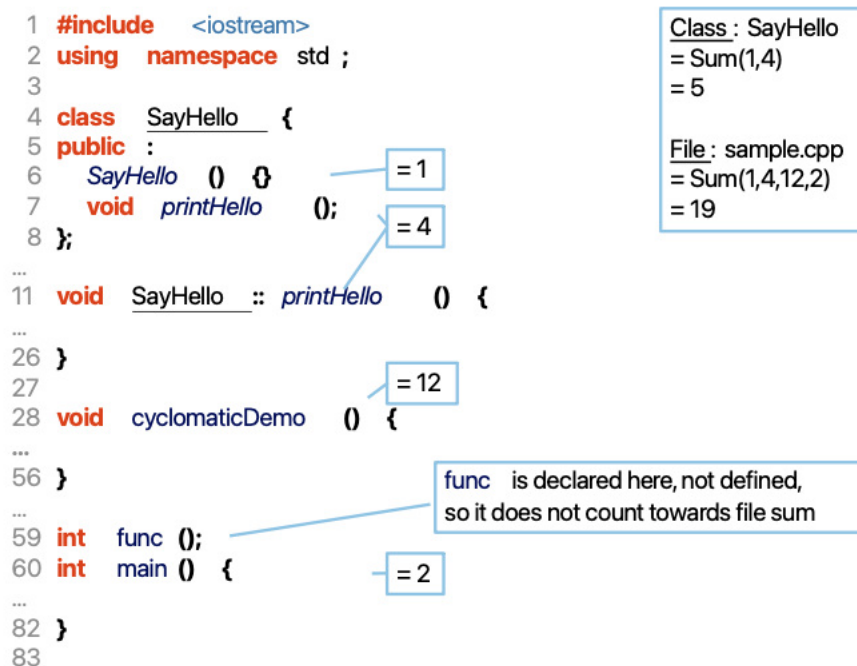
Sum Strict Cyclomatic Complexity

API ID: SumCyclomaticStrict

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Architectures, Classes, Files, Functions, Modules, Packages, Project, Subprograms

Sum of strict cyclomatic complexity of all nested functions or methods.



```

1 #include <iostream>
2 using namespace std ;
3
4 class SayHello {
5 public :
6     SayHello () {}
7     void printHello () ;
8 };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {
...
56 }
...
59 int func () ;
60 int main () {
...
82 }
83

```

Class : SayHello
= Sum(1,4)
= 5

File : sample.cpp
= Sum(1,4,12,2)
= 19

func is declared here, not defined,
so it does not count towards file sum

Targets By Language:

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Project, File, Method, Module, Class, Struct
- **C#:** Project, File, Class, Struct, Method
- **C++:** Project, File, Class, Struct, Union
- **Fortran:** Project, File, Module, Function, Program, Subroutine
- **Java:** Project, File, Class, Interface, Method
- **Jovial:** Project, File
- **Pascal:** Project, File, Class, Interface, Compunit, Function, Procedure
- **Python:** Project, File, Class
- **Rust:** Project, File
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- The cyclomatic metric used is configured from "Project Configuration/Metrics/Complexity". This metric results from having "Count each decision in a multi-way-decision structure" on and having "Count logical conjunctions (OR) and logical and (AND)" on. Checking "Show all available cyclomatic complexity metric variants" will also enable the metric.

Sum Strict Modified Cyclomatic Complexity

API ID: SumCyclomaticStrictModified

Languages: Ada, Basic, C#, C++, Fortran, Java, Jovial, Pascal, Python, Rust, Web

Targets: Architectures, Classes, Files, Functions, Modules, Packages, Project, Subprograms

Sum of strict modified cyclomatic complexity of all nested functions or methods.

```

1  #include <iostream>
2  using namespace std ;
3
4  class SayHello {
5  public :
6      SayHello () {}           = 1
7      void printHello () ;     = 3
8  };
...
11 void SayHello :: printHello () {
...
26 }
27
28 void cyclomaticDemo () {     = 10
...
56 }
...
59 int func () ;
60 int main () {               = 2
...
82 }
83

```

Class : SayHello
 = Sum(1,3)
 = 4

 File : sample.cpp
 = Sum(1,3,10,2)
 = 16

func is declared here, not defined,
 so it does not count towards file sum

Targets By Language:

- **Ada:** Project, File, Type, Entry, Function, Package, Procedure, Protected, Task
- **Basic:** Project, File, Method, Module, Class, Struct
- **C#:** Project, File, Class, Struct, Method
- **C++:** Project, File, Class, Struct, Union
- **Fortran:** Project, File, Module, Function, Program, Subroutine
- **Java:** Project, File, Class, Interface, Method
- **Jovial:** Project, File
- **Pascal:** Project, File, Class, Interface, Compunit, Function, Procedure
- **Python:** Project, File, Class
- **Rust:** Project, File
- **Web:** Project, File, PHP Class, PHP Interface

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- The cyclomatic metric used is configured from "Project Configuration/Metrics/Complexity". This metric results from having "Count each decision in a multi-way-decision structure" off and having "Count logical conjunctions (OR) and logical and (AND)" on. Checking "Show all available cyclomatic complexity metric variants" will also enable the metric.

Sum Strict Modified Essential Complexity

API ID: SumEssentialStrictModified

Languages: Ada

Targets: Architectures, Files, Packages, Project

Sum of strict modified essential complexity of all nested functions or methods.

Targets By Language:

- **Ada:** Project, File, Package

Configuration:

- For projects and architectures, this metric must also be enabled from "Project Configuration/Metrics/Projects and Architectures" with the "Show aggregated function metrics" option.
- This metric can be enabled from "Project Configuration/Metrics/Complexity" with the "Essential" option.