

Average Lines
Average Blank Lines
Average Blank Lines (Includes Inactive)
Average Code Lines
Average Code Lines (Includes Inactive)
Average Comment Lines
Average Comment Lines (Includes Inactive)
Average Cyclomatic Complexity
Average Modified Cyclomatic Complexity
Average Strict Cyclomatic Complexity
Average Strict Modified Cyclomatic Complexity
Average Essential Complexity
Average Strict Modified Essential Complexity
Base Classes
Coupled Classes
Coupled Classes Modified
Derived Classes
Classes
Class Methods
Class Variables
Executable Units
Files
Code Files
Header Files
Functions
Instance Methods
Instance Variables
Internal Instance Variables
Private Instance Variables
Protected Instance Variables
Protected Internal Instance Variables
Public Instance Variables
Methods
All Methods
Const Methods
Default Methods
Friend Methods
Internal Methods
Private Methods
Protected Methods
Protected Internal Methods

Public Methods
Strict Private Methods
Strict Published Methods
Modules
Program Units
Properties
Auto-Implemented Properties
Subprograms
Inputs
Lines
Blank Lines
Blank Lines (HTML)
Blank Lines (JavaScript)
Blank Lines (PHP)
Blank Lines (Includes Inactive)
Code Lines
Declarative Code Lines
Executable Code Lines
Code Lines (JavaScript)
Code Lines (PHP)
Code Lines (Includes Inactive)
Comment Lines
Comment Lines (HTML)
Comment Lines (JavaScript)
Comment Lines (PHP)
Comment Lines (Includes Inactive)
Lines (HTML)
Inactive Lines
Lines (JavaScript)
Lines (PHP)
Preprocessor Lines
Outputs
Coupled Packages
Paths
Paths Log(x)
Semicolons
Statements
Declarative Statements
Declarative Statements (Javascript)
Declarative Statements (PHP)
Empty Statements
Executable Statements
Executable Statements (JavaScript)
Executable Statements (PHP)
Cyclomatic Complexity
Modified Cyclomatic Complexity

Strict Cyclomatic Complexity
Strict Modified Cyclomatic Complexity
Essential Complexity
Strict Modified Essential Complexity
Knots
Max Cyclomatic Complexity
Max Modified Cyclomatic Complexity
Max Strict Cyclomatic Complexity
Max Strict Modified Cyclomatic Complexity
Max Essential Complexity
Max Essential Knots
Max Strict Modified Essential Complexity
Max Inheritance Tree
Max Nesting
Min Essential Knots
Percent Lack Of Cohesion
Percent Lack Of Cohesion Modified
Comment to Code Ratio
Sum Cyclomatic Complexity
Sum Modified Cyclomatic Complexity
Sum Strict Cyclomatic Complexity
Sum Strict Modified Cyclomatic Complexity
Sum Essential Complexity
Sum Strict Modified Essential Complexity

```

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

```

= 1

= 16

= 29

= 23

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


```

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
 $= \text{Average}(0,1)$
 $= 0.5 = 1$

File: sample.cpp
 $= \text{Average}(0,1,2,4)$
 $= 1.75 = 2$

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

```

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
 $= \text{Average}(0,2)$
 $= 1$

File: sample.cpp
 $= \text{Average}(0,2,2,5)$
 $= 2.25 = 2$

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

```

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

```

= 1

= 11

= 27

= 13

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

```

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

```

= 1

= 14

= 27

= 17

Class : SayHello
 = Average(1,14)
 = 7.5 = 8

File : sample.cpp
 = Average(1,14,27,17)
 = 14.75 = 15

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

```

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
 $= \text{Average}(0,1)$
 $= 0.5 = 1$

File: sample.cpp
 $= \text{Average}(0,1,0,0)$
 $= 0.25 = 0$

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

```

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
 $= \text{Average}(0,2)$
 $= 1$

File: sample.cpp
 $= \text{Average}(0,2,0,0)$
 $= 0.5 = 1$

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

```

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

```

= 1

= 4

= 10

= 2

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

```

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

```

= 1

= 3

= 8

= 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

```

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

```

= 1

= 4

= 12

= 2

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

```

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

```

= 1

= 3

= 10

= 2

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


```

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

```

= 1

= 3

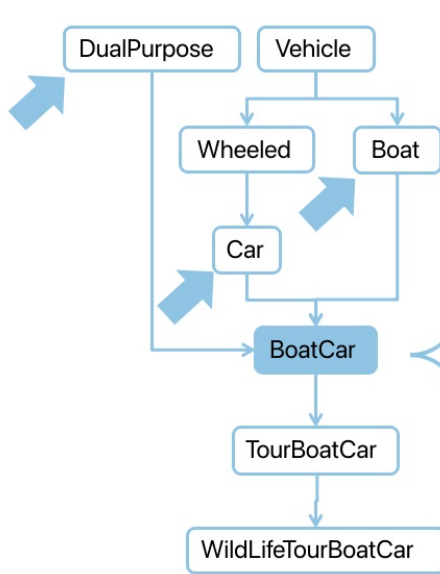
= 1

= 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



```

34 class BoatCar : private Car public Boat, protected DualPurpose {
35 public :
36 // Public Instance Function
37 BoatCar () : Car (4), Boat (), mInWater (false ), mColor ("Blue" ) {}
38 virtual int passengers () const { return 4; }
39
40 static int numRegistered () { return sRegistered ; }
41
42 bool mInWater ; // Public Instance Variable
43
44 protected :
45
46 void toggleInWater (bool inWater ) { mInWater = inWater ; }
47 char * mColor ; // Protected Instance Variable
48 friend void init () {}
49
50 static int sRegistered ;
51 static double calcSpeed (double distance , double time ) {
52     return distance / time ;
53 }
54
55 private :
56 int mMaxPassengers ;
57 void travel () {}
58 };

```

= 3

= 3

Backward references don't count

```
57 class Snake {
58 public :
59 void eatFrog (Frog f) {
60     if (!f.swimming())
61         hunger --;
62 }
63 private :
64 int hunger ;
65 };
```

References to nested class don't count

References to base class don't count

```
7 class Amphibian {
8 public :
9     typedef Bird * bird_ptr ;
10
11 void eatenbybird () {
12     mBird ->eat (this);
13 }
14 private :
15 bird_ptr mBird ;
16 };
```

```
32 class Frog : public Amphibian {
33 public :
34 bool swimming () {
35     if (Water::temp() > 50)
36         return 1;
37     return 0;
38 }
39 void eatFly () {
40     Fly edible ;
41     edible .getEaten () ;
42 }
43 private :
44 class HopCalculator {
45 public :
46     int calculateHops () { return 1;}
47 };
48
49 Toad mCousin ;
50
51 void hop () {
52     HopCalculator (). calculateHops () ;
53     Amphibian::eatenbybird () ;
54 }
55 };
```

Inherited functions don't count,
even when called in class.

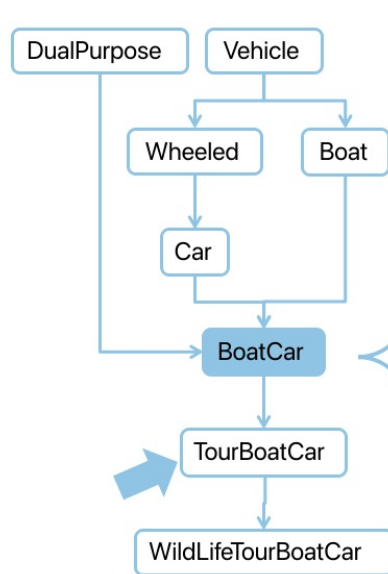
```
22 class Water {
23 public :
24     static int temp ()
25     { return 60; }
26 };
```

```
27 class Fly {
28 public :
29     void getEaten () {}
30 };
```

These count as 1 since they
reference the same class.

```
18 class Toad :
19     public Amphibian {
20 };
```

```
2 class Bird {
3 public :
4     void eat (Amphibian *) {}
5 };
```

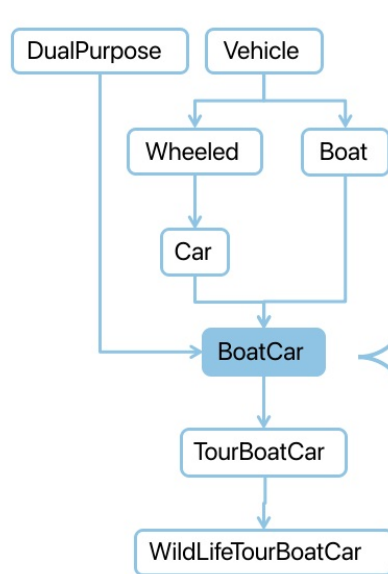


```

34 class BoatCar : private Car, public Boat, protected DualPurpose {
35 public :
36     // Public Instance Function
37     BoatCar () : Car (4), Boat (), mInWater (false ), mColor ("Blue" ) {}
38     virtual int passengers () const { return 4; }
39
40     static int numRegistered () { return sRegistered ; }
41
42     bool mInWater ; // Public Instance Variable
43
44 protected :
45
46     void toggleInWater (bool inWater ) { mInWater = inWater ; }
47     char * mColor ; // Protected Instance Variable
48     friend void init () {}
49
50     static int sRegistered ;
51     static double calcSpeed (double distance , double time ) {
52         return distance / time ;
53     }
54
55 private :
56     int mMaxPassengers ;
57     void travel () {}
58 };

```

= 1

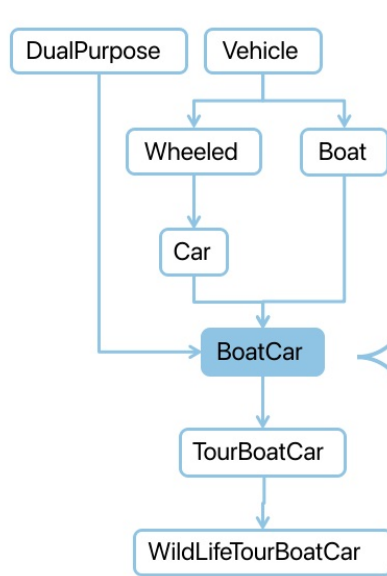


```

34 class BoatCar : private Car, public Boat, protected DualPurpose {
35 public :
36 // Public Instance Function
37 BoatCar () : Car (4), Boat (), mInWater (false ), mColor ("Blue" ) {}
38 virtual int passengers () const { return 4; }
39
40 static int numRegistered () { return sRegistered ; }
41
42 bool mInWater ; // Public Instance Variable
43
44 protected :
45
46 void toggleInWater (bool inWater ) { mInWater = inWater ; }
47 char * mColor ; // Protected Instance Variable
48 friend void init () {}
49
50 static int sRegistered ;
51 static double calcSpeed (double distance , double time ) {
52 return distance / time ;
53 }
54
55 private :
56 int mMaxPassengers ;
57 void travel () {}
58 };

```

= 2

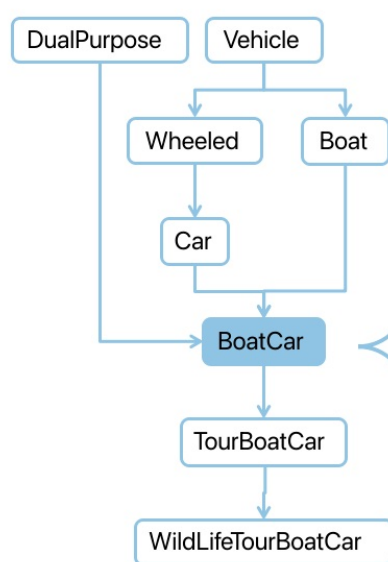


```

34 class BoatCar : private Car, public Boat, protected DualPurpose {
35 public :
36 // Public Instance Function
37 BoatCar () : Car (4), Boat (), mInWater (false ), mColor ("Blue" ) {}
38 virtual int passengers () const { return 4; }
39
40 static int numRegistered () { return sRegistered ; }
41
42 bool mInWater ; // Public Instance Variable
43
44 protected :
45
46 void toggleInWater (bool inWater ) { mInWater = inWater ; }
47 char * mColor ; // Protected Instance Variable
48 friend void init () {}
49
50 static int sRegistered ;
51 static double calcSpeed (double distance , double time ) {
52     return distance / time ;
53 }
54
55 private :
56 int mMaxPassengers ;
57 void travel () {}
58 };

```

= 1



```

34 class BoatCar : private Car, public Boat, protected DualPurpose {
35 public :
36 // Public Instance Function
37 BoatCar () : Car (4), Boat (), mInWater (false), mColor ("Blue" ) {}
38 virtual int passengers () const { return 4; }
39
40 static int numRegistered () { return sRegistered ; }
41
42 bool mInWater ; // Public Instance Variable
43
44 protected :
45
46 void toggleInWater (bool inWater ) { mInWater = inWater ; }
47 char * mColor ; // Protected Instance Variable
48 friend void init () {}
49
50 static int sRegistered ;
51 static double calcSpeed (double distance , double time ) {
52     return distance / time ;
53 }
54
55 private :
56 int mMaxPassengers ;
57 void travel () {}
58 };

```

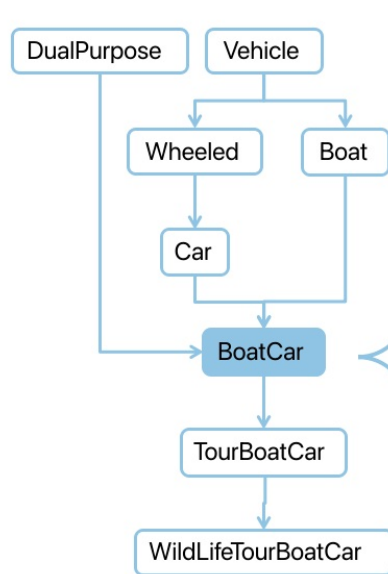
Strict includes implicit methods:

```

BoatCar (const BoatCar &)
BoatCar (BoatCar &&)
~BoatCar ()
operator =(const BoatCar &)
operator =(BoatCar &&)

```

= 4 (Fuzzy)
= 9 (Strict)

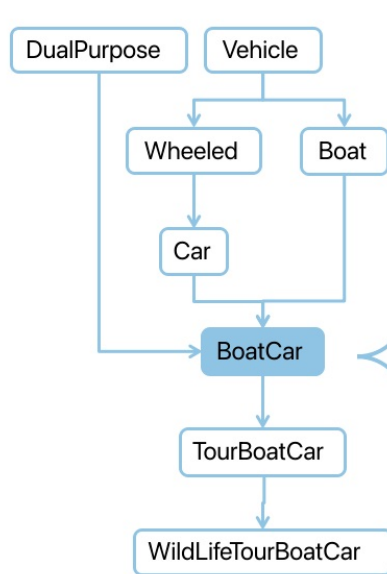


```

34 class BoatCar : private Car, public Boat, protected DualPurpose {
35 public :
36 // Public Instance Function
37 BoatCar () : Car (4), Boat (), mInWater (false ), mColor ("Blue" ) {}
38 virtual int passengers () const { return 4; }
39
40 static int numRegistered () { return sRegistered ; }
41
42 bool mInWater ; // Public Instance Variable
43
44 protected :
45
46 void toggleInWater (bool inWater ) { mInWater = inWater ; }
47 char * mColor ; // Protected Instance Variable
48 friend void init () {}
49
50 static int sRegistered ;
51 static double calcSpeed (double distance , double time ) {
52     return distance / time ;
53 }
54
55 private :
56 int mMaxPassengers ;
57 void travel () {}
58 };

```

= 3

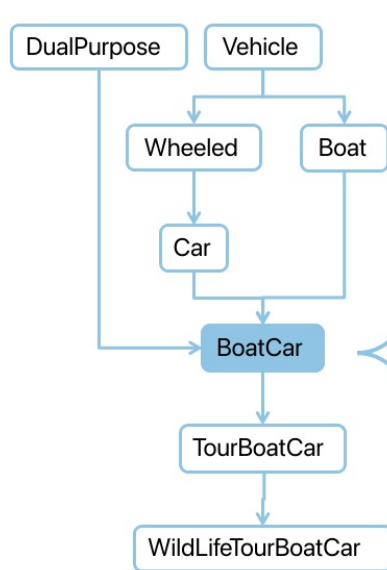


```

34 class BoatCar : private Car, public Boat, protected DualPurpose {
35 public :
36     // Public Instance Function
37     BoatCar () : Car (4), Boat (), mInWater (false ), mColor ("Blue" ) {}
38     virtual int passengers () const { return 4; }
39
40     static int numRegistered () { return sRegistered ; }
41
42     bool mInWater ; // Public Instance Variable
43
44 protected :
45
46     void toggleInWater (bool inWater ) { mInWater = inWater ; }
47     char * mColor ; // Protected Instance Variable
48     friend void init () {}
49
50     static int sRegistered ;
51     static double calcSpeed (double distance , double time ) {
52         return distance / time ;
53     }
54
55 private :
56     int mMaxPassengers ;
57     void travel () {}
58 };

```

= 1

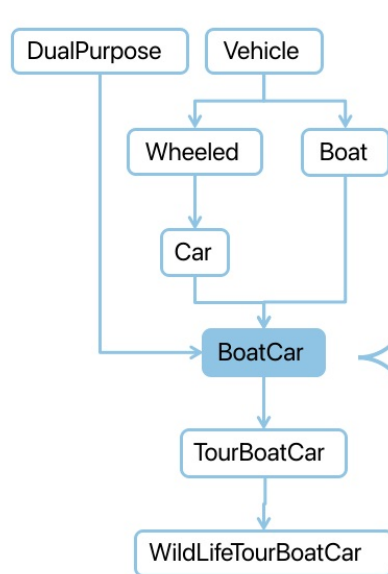


```

34 class BoatCar : private Car, public Boat, protected DualPurpose {
35 public :
36     // Public Instance Function
37     BoatCar () : Car (4), Boat (), mInWater (false ), mColor ("Blue" ) {}
38     virtual int passengers () const { return 4; }
39
40     static int numRegistered () { return sRegistered ; }
41
42     bool mInWater ; // Public Instance Variable
43
44 protected :
45
46     void toggleInWater (bool inWater ) { mInWater = inWater ; }
47     char * mColor ; // Protected Instance Variable
48     friend void init () {}
49
50     static int sRegistered ;
51     static double calcSpeed (double distance , double time ) {
52         return distance / time ;
53     }
54
55 private :
56     int mMaxPassengers ;
57     void travel () {}
58 };

```

= 1

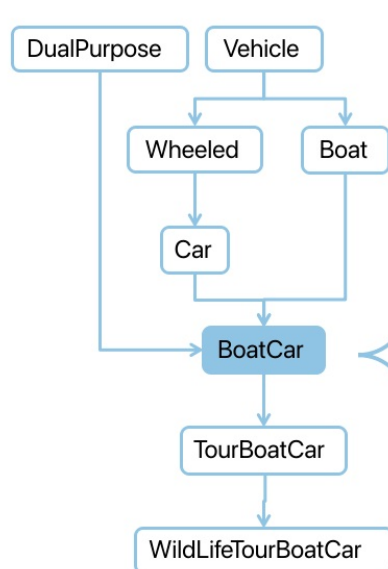


```

34 class BoatCar : private Car, public Boat, protected DualPurpose {
35 public :
36     //Public Instance Function
37     BoatCar () : Car (4), Boat (), mInWater (false ), mColor ( "Blue" ) {}
38     virtual int passengers () const { return 4; }
39
40     static int numRegistered () { return sRegistered ; }
41
42     bool mInWater ; //Public Instance Variable
43
44 protected :
45
46     void toggleInWater (bool inWater ) { mInWater = inWater ; }
47     char * mColor ; //Protected Instance Variable
48     friend void init () {}
49
50     static int sRegistered ;
51     static double calcSpeed (double distance , double time ) {
52         return distance / time ;
53     }
54
55 private :
56     int mMaxPassengers ;
57     void travel () {}
58 };

```

= 1

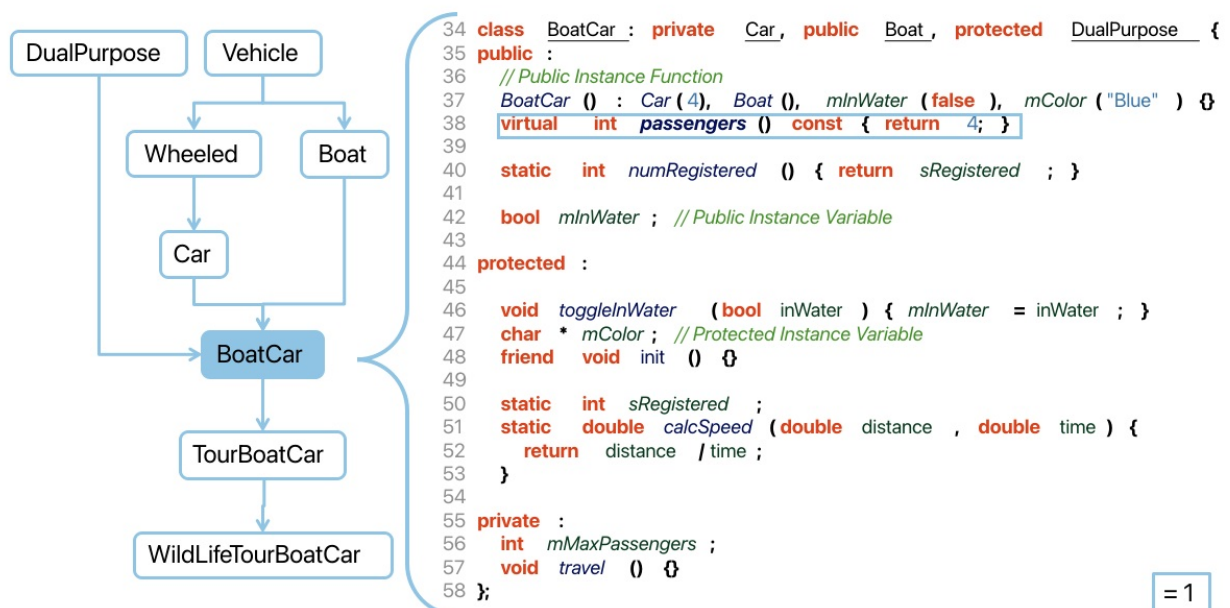
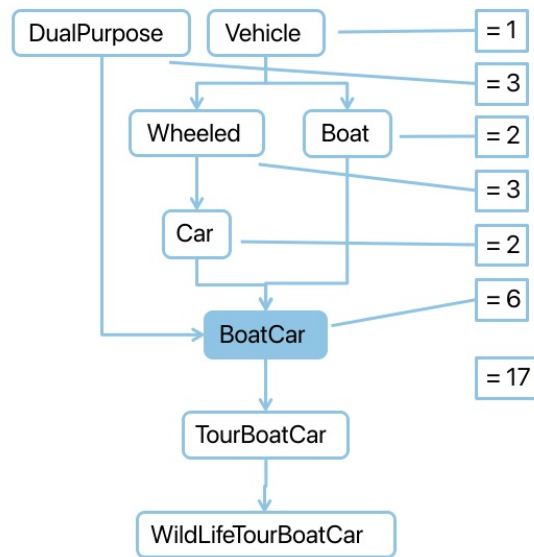


```

34 class BoatCar : private Car, public Boat, protected DualPurpose {
35 public :
36     // Public Instance Function
37     BoatCar () : Car (4), Boat (), mInWater (false ), mColor ( "Blue" ) {}
38     virtual int passengers () const { return 4; }
39
40     static int numRegistered () { return sRegistered ; }
41
42     bool mInWater ; // Public Instance Variable
43
44 protected :
45
46     void toggleInWater (bool inWater ) { mInWater = inWater ; }
47     char * mColor ; // Protected Instance Variable
48     friend void init () {}
49
50     static int sRegistered ;
51     static double calcSpeed (double distance , double time ) {
52         return distance / time ;
53     }
54
55 private :
56     int mMaxPassengers ;
57     void travel () {}
58 };

```

= 6



```

1  class CohesionClass {
2  public :
3      void func1 () {
4          ...
5          ...
6          ...
7          ...
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
29     int mVar1 ;
30     char * mVar2 ;
31     static int sNumObjs ;
32 };

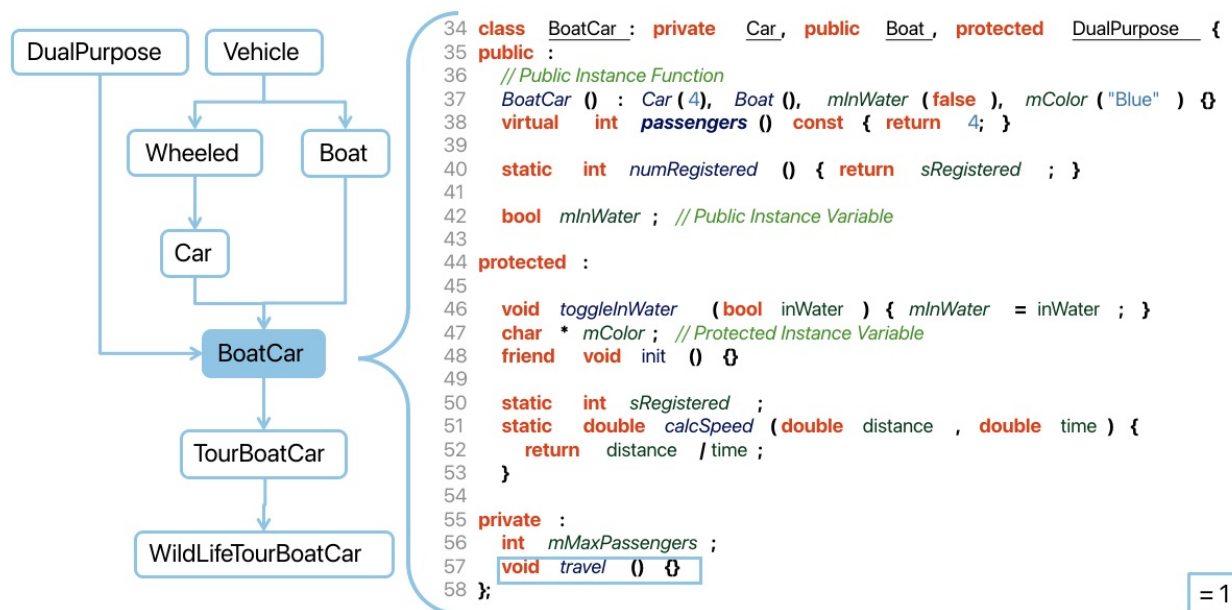
```

```

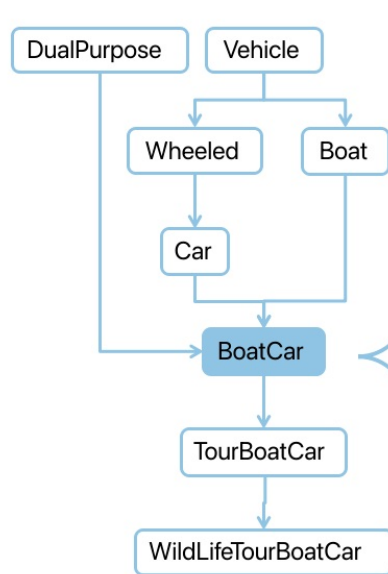
1  class FriendDemo {
2  friend class CohesionClass ;
3
4  friend void init () ;
5
6  };

```

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



= 1

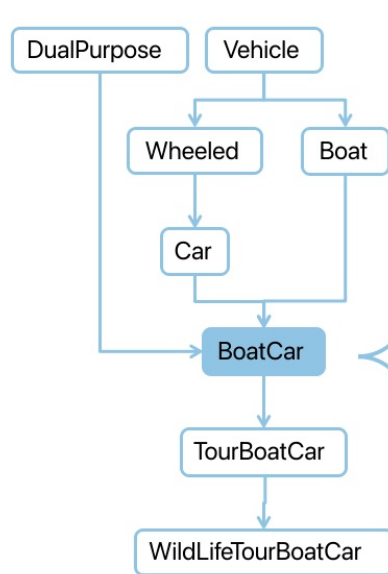


```

34 class BoatCar : private Car, public Boat, protected DualPurpose {
35 public :
36 //Public Instance Function
37 BoatCar () : Car (4), Boat (), mInWater ( false ), mColor ( "Blue" ) {}
38 virtual int passengers () const { return 4; }
39
40 static int numRegistered () { return sRegistered ; }
41
42 bool mInWater ; // Public Instance Variable
43
44 protected :
45
46 void toggleInWater ( bool inWater ) { mInWater = inWater ; }
47 char * mColor ; //Protected Instance Variable
48 friend void init () {}
49
50 static int sRegistered ;
51 static double calcSpeed ( double distance , double time ) {
52     return distance / time ;
53 }
54
55 private :
56 int mMaxPassengers ;
57 void travel () {}
58 };

```

= 2



```

34 class BoatCar : private Car, public Boat, protected DualPurpose {
35 public :
36 //Public Instance Function
37 BoatCar () : Car (4), Boat (), mInWater (false ), mColor ( "Blue" ) {}
38 virtual int passengers () const { return 4; }
39
40 static int numRegistered () { return sRegistered ; }
41
42 bool mInWater ; // Public Instance Variable
43
44 protected :
45
46 void toggleInWater ( bool inWater ) { mInWater = inWater ; }
47 char * mColor ; //Protected Instance Variable
48 friend void init () {}
49
50 static int sRegistered ;
51 static double calcSpeed ( double distance , double time ) {
52     return distance / time ;
53 }
54
55 private :
56 int mMaxPassengers ;
57 void travel () {}
58 };

```

= 3


```

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 (0);
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

Start_Line = 11	—	<pre> 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 } </pre>
End_Line = 26	—	<pre> = End_Line - Start_Line + 1 = 26 - 11 + 1 = 16 </pre>

					= not (Code Comment Preprocessor Inactive) = 1	
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 }

Inactive blank lines do not count

					= not (Code Comment Preprocessor) = 2	
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 }

[illegible]

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 }

= Code && Declarative
= 2

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 }

```

= Code && Executable
= 8

```
= Code || Preprocessor
= 14
```

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 ; //
24	#endif					
25						
26		}				

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 }

= Comment && not(Inactive)
= 1

Inactive comment lines do not count

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 }

= Comment
= 2

Code	Comment	Preprocessor	Declarative	Executable	Inactive
✓			✓		
✓				✓	
✓				✓	
✓				✓	
✓				✓	
✓	✓			✓	
✓			✓	✓	
✓				✓	
✓					
		✓			
✓	✓			✓	
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					

```

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 }

```

= Inactive
= 2

= Preprocessor
= 2

Code	Comment	Preprocessor	Declarative	Executable	Inactive
✓		✓			
✓			✓		
✓			✓		
✓			✓		
✓			✓		
✓	✓		✓		
✓			✓	✓	
✓			✓	✓	
✓				✓	
✓					
<input checked="" type="checkbox"/>		✓			
✓	✓			✓	
<input checked="" type="checkbox"/>					
✓					

```

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 }

```

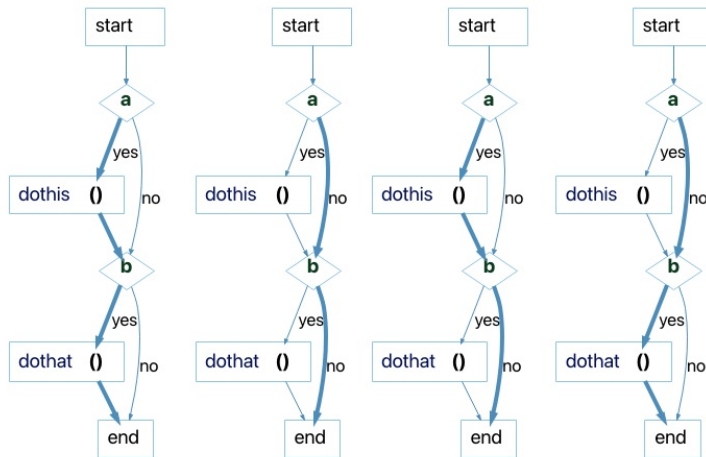
```

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

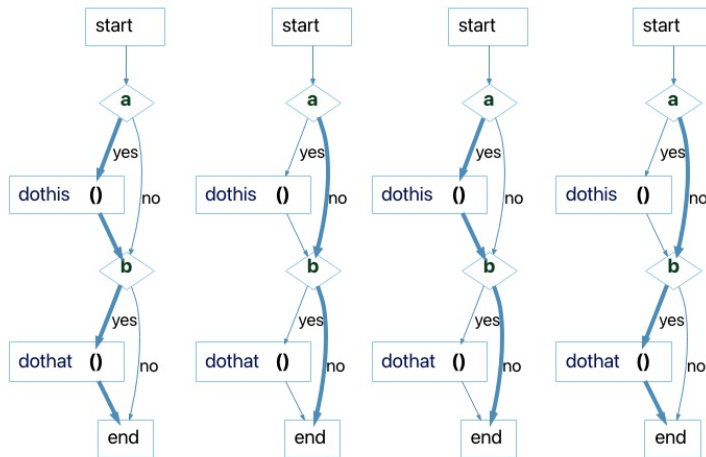


= 4

```

5 void pathDemo () {
6   if (a)
7     dothis ();
8   if (b)
9     dothat ();
10 }

```

$$= \log_{10}(4) \\ = 1$$

```

5 void pathDemo () {
6   if (a)
7     dothis ();
8   if (b)
9     dothat ();
10 }
  
```

```

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

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

```

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 }

```

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
1	0	0	1
1	0	0	1
0	0	0	0
0	0	0	0
6 (4)	6	1	11

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

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

```

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 }

```

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
0	1	0	1
1 (0)	0	0	0
0	1	0	1
1	0	0	1

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

1	0	0	1
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

```
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 }
```

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

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

1	0	0	1
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

```
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 }
```

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	1
0	1	0	1
1 (0)	0	0	0
0	1	0	1
1	0	0	1
1	0	0	1
0	0	0	0
0	0	0	0
6 (4)	6	1	11

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

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 << <<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 }

= decision points + 1
= 9 + 1
= 10

SWITCH	CASE	CATCH	DO	FOR	IF	?	WHILE	AND	OR
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Modified	Modified	Modified	Modified	Modified	Modified	Modified	Modified	Modified	Modified
Not Modified	Not Modified	Not Modified	Not Modified	Not Modified	Not Modified	Not Modified	Not Modified	Not Modified	Not Modified
1	3	1	1	1	1	1	1	1	1

```

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 }

```

= decision points + 1
 = 7 + 1
 = 8

	SWITCH											
	CASE	CATCH	DO	FOR	IF	?	WHILE	AND	OR			
✓	✓			✓		✓	✓	✓	✓			28 void cyclomaticDemo () {
✓	✓											29 bool a = true , b = true , c = true ;
✓	✓											30
Modified												31 if (a (b && c)) {
Not Modified												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 }
1	3	1	1	1	1	1	1	1	1			56 }

= decision points + 1
= 11 + 1
= 12

	SWITCH	CASE	CATCH	DO	FOR	IF	?	WHILE	AND	OR	
1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	28 void cyclomaticDemo () {
2											29 bool a = true , b = true , c = true ;
3											30
4											31 if (a (b && c)) {
5											32 while (a ? b : c) {
6											33 for (int i = 0; i < 10; i++) {
7											34 switch (i) {
8											35 case 1:
9											36 case 2:
10											37 cout << i << endl ;
11											38 break ;
12											39 case 5:
13											40 break ;
14											41 default :
15											42 cout << i << endl ;
16											43 }
17											44 }
18											45 }
19											46 } else {
20											47 try {
21											48 do {
22											49 cout << a << b << c << endl ;
23											50 } while (a);
24											51 }
25											52 catch (...){
26											53 }
27											54 }
28											55 }
29											56 }

= decision points + 1
= 9 + 1
= 10

```

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 ();
15             else if (j)
16                 dosomething ();
17             else if (k)
18                 dosomething ();
19             else {}
20
21         }
22     }
23 }
24 }

```

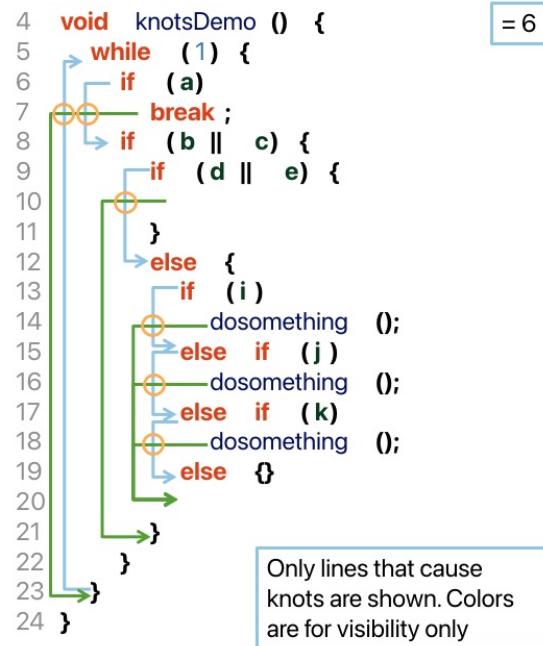


```

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

```

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



```

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 () {}` is annotated with `= 1`.
- Line 7: `void printHello () ;` is annotated with `= 4`.
- Line 28: `void cyclomaticDemo () {` is annotated with `= 10`.
- Line 60: `int main () {` is annotated with `= 2`.
- A box points to line 59: `func () ;` with the text: "func is declared here, not defined, so it does not count towards file max".

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

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

```

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 code:

- Line 6: `SayHello () {}` is annotated with `= 1`.
- Line 7: `void printHello () ;` is annotated with `= 3`.
- Line 28: `void cyclomaticDemo () {` is annotated with `= 8`.
- Line 60: `int main () {` is annotated with `= 2`.

Summary box:

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

```

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 () ;` → = 4
- Line 28: `void cyclomaticDemo () {` → = 12
- Line 60: `int main () {` → = 2

Summary of complexity values:

- 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

```

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:

- A box containing "= 1" points to the `SayHello () {}` line (line 6).
- A box containing "= 3" points to the `void printHello () ;` line (line 7).
- A box containing "= 10" points to the `void cyclomaticDemo () {` line (line 28).
- A box containing "= 2" points to the `int main () {` line (line 60).
- A box containing "func is declared here, not defined, so it does not count towards file max" points to the `int func ();` line (line 59).

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

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


```

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

```

= 1

= 3

= 1

= 1

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

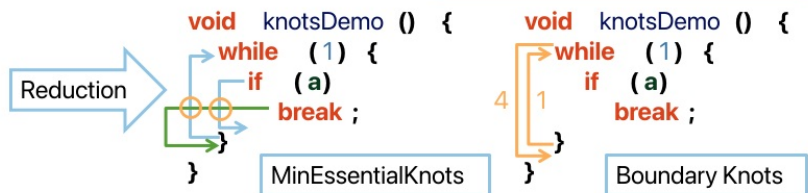
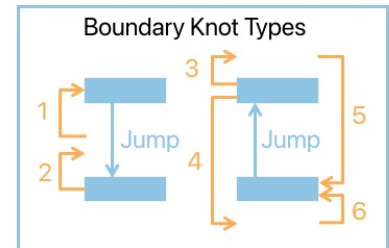
File : sample.cpp
= Max(1,3,1,1)
= 3

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

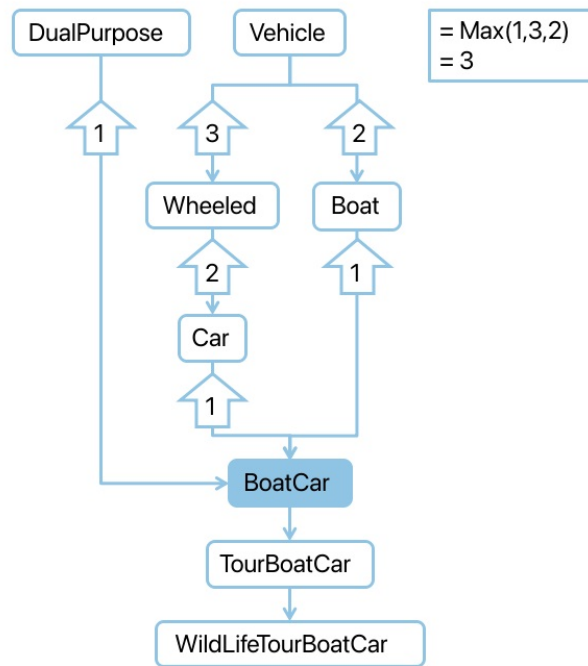
```

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 ();
15      else if (j)
16        dosomething ();
17      else if (k)
18        dosomething ();
19      else {}
20    }
21  }
22 }
23 }
24 }

```



$$\begin{aligned}
 &= \text{MinEssentialKnots} + (\text{Boundary Knots} / 2) \\
 &= 2 + (2/2) \\
 &= 3
 \end{aligned}$$



Nesting

```
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 }
```

= 4

```

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 ();
16       else if (j)
17         dosomething ();
18       else if (k)
19         dosomething ();
20       else {}
21     }
22   }
23 }
24 }

```

Reduction

```

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

```

= Knots
= 2

```

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%		

Comment
Code

= CountLineComment / CodeLineCode
= 1 / 11
= 0.09

```
✓ 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 }
```

```

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

```

= 1

= 4

= 10

= 2

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


```

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

```

= 1

= 3

= 8

= 2

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

```

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

```

= 1

= 4

= 12

= 2

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

```

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

```

= 1

= 3

= 10

= 2

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

```

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 11: `void SayHello :: printHello () {` → = 1
- Line 28: `void cyclomaticDemo () {` → = 1
- Line 60: `int main () {` → = 1

Summary of complexity values:

- 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