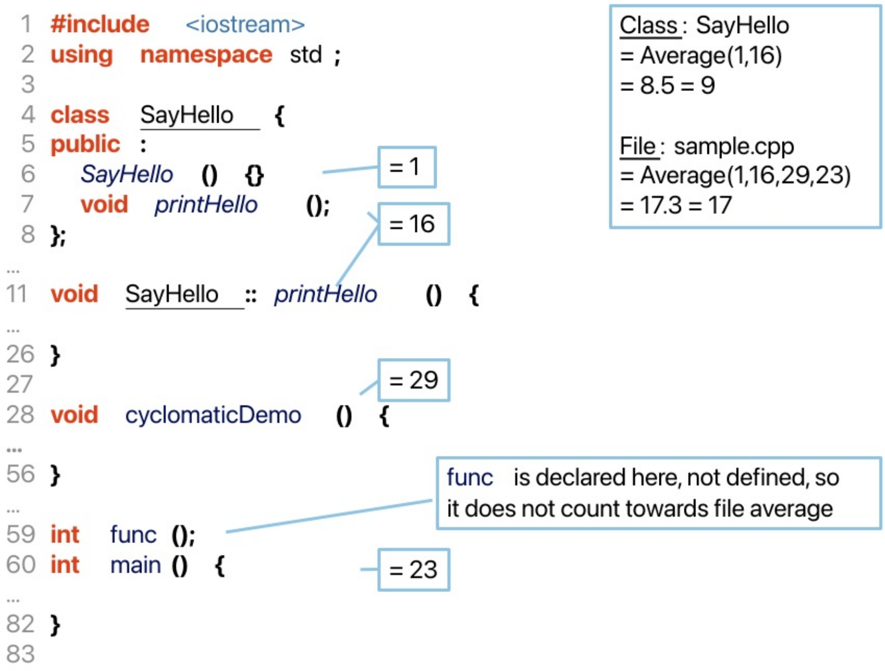


```
#include <iostream>
using namespace std ;

class SayHello {
public :
   SayHello () {}
   void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

= 1

= 16

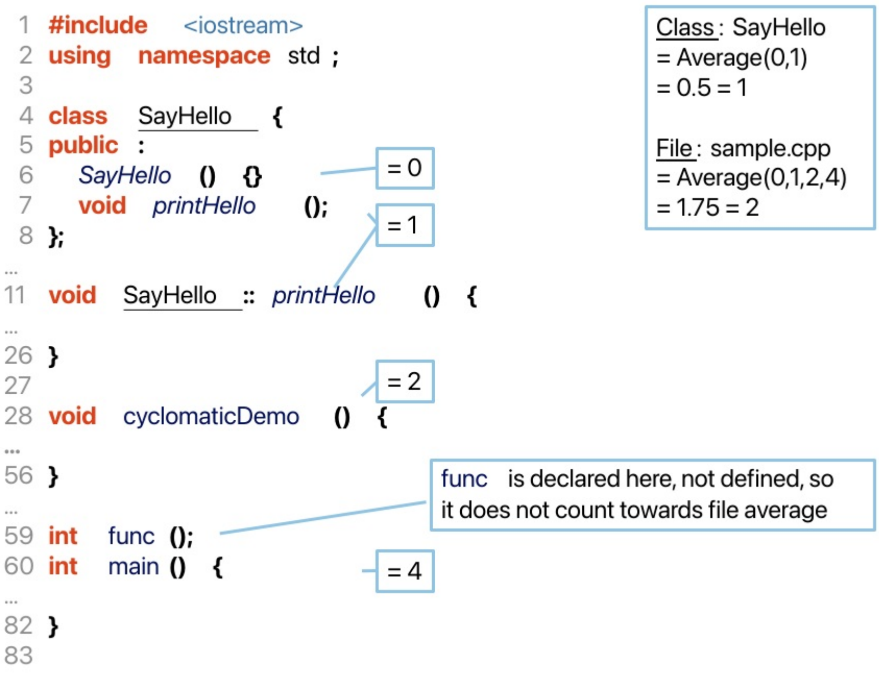
= 29

= 23

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

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

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

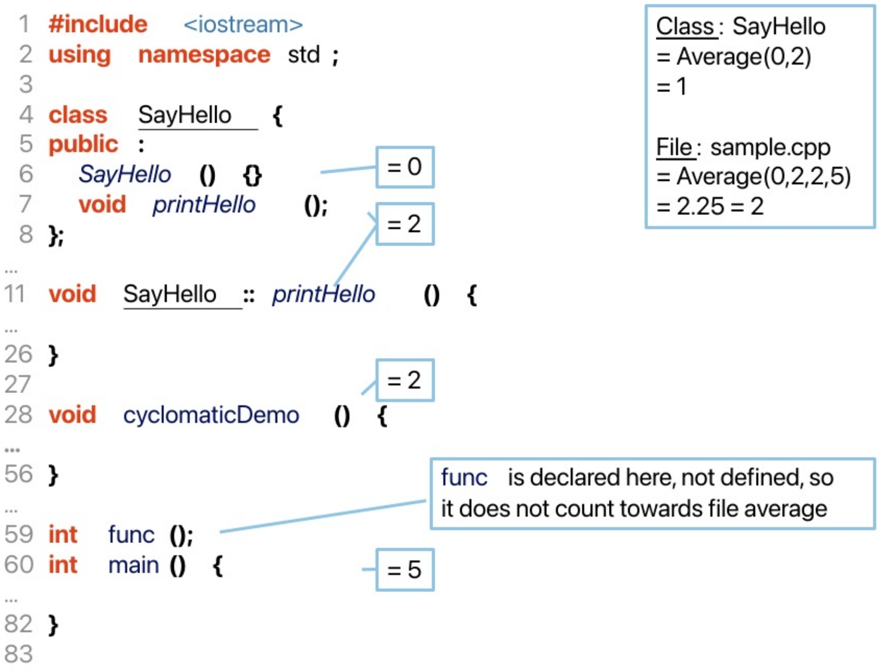


```
#include <iostream>
using namespace std ;

class SayHello {
public :
   SayHello () {}
   void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

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

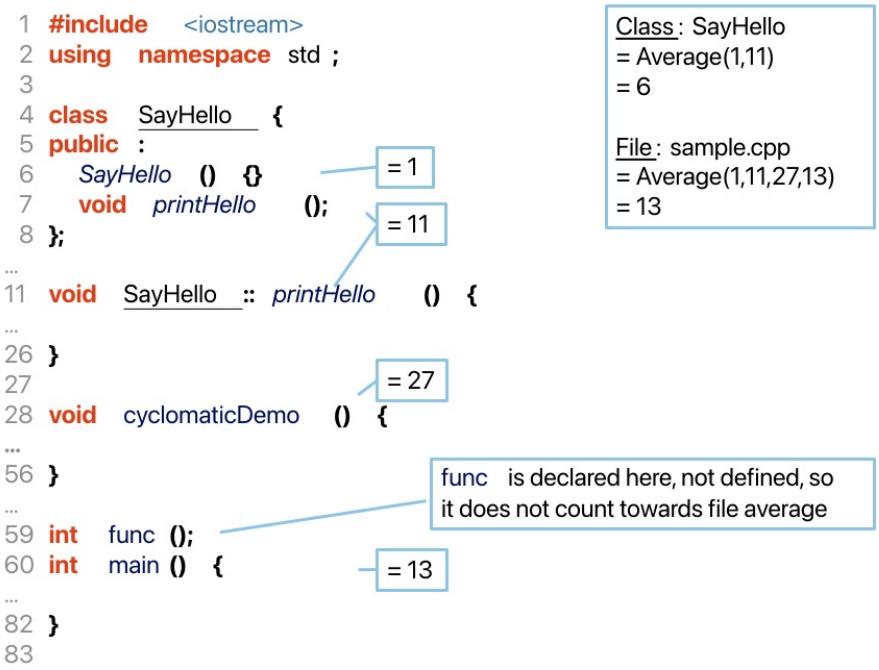


```
#include <iostream>
using namespace std ;

class SayHello {
public :
   SayHello () {}
   void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

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

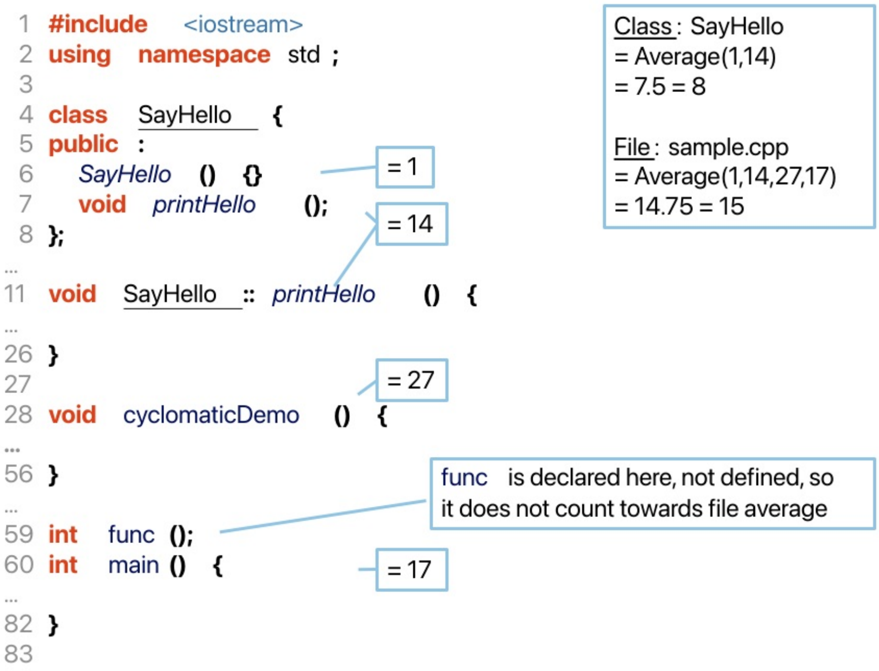


```
#include <iostream>
using namespace std ;

class SayHello {
public :
   SayHello () {}
   void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

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



```
#include <iostream>
using namespace std ;

class SayHello {
public :
    SayHello () {}
    void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

= 1

= 14

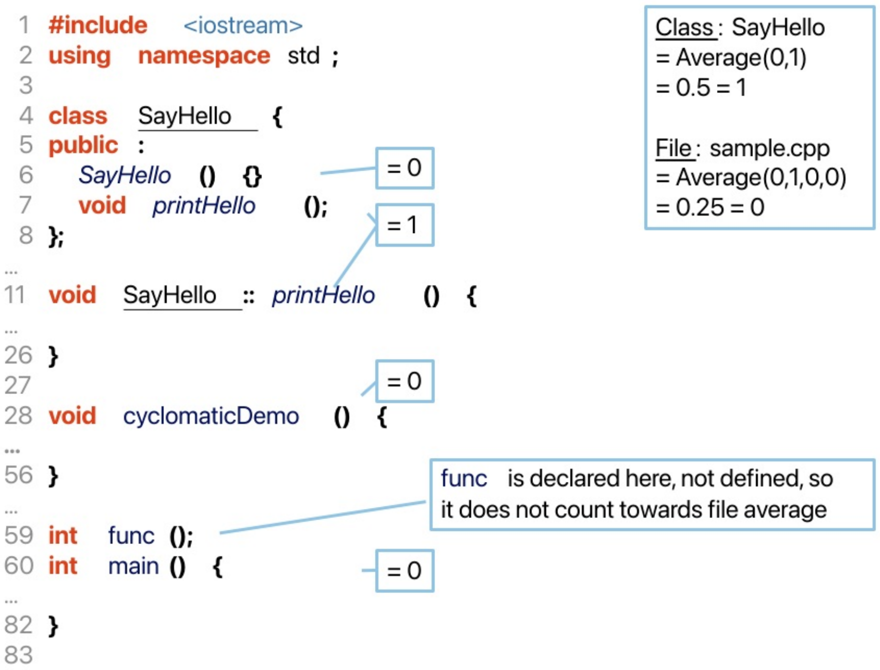
= 27

= 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



```
#include   <iostream>
using   namespace   std ;

class   SayHello   {
public   :
   SayHello   ()   {}
   void   printHello      ();
};
...
void   SayHello   ::  printHello      ()   {
...
}

void   cyclomaticDemo    ()   {
...
}
...
int   func  ();
int   main  ()   {
...
}

```

= 0

= 1

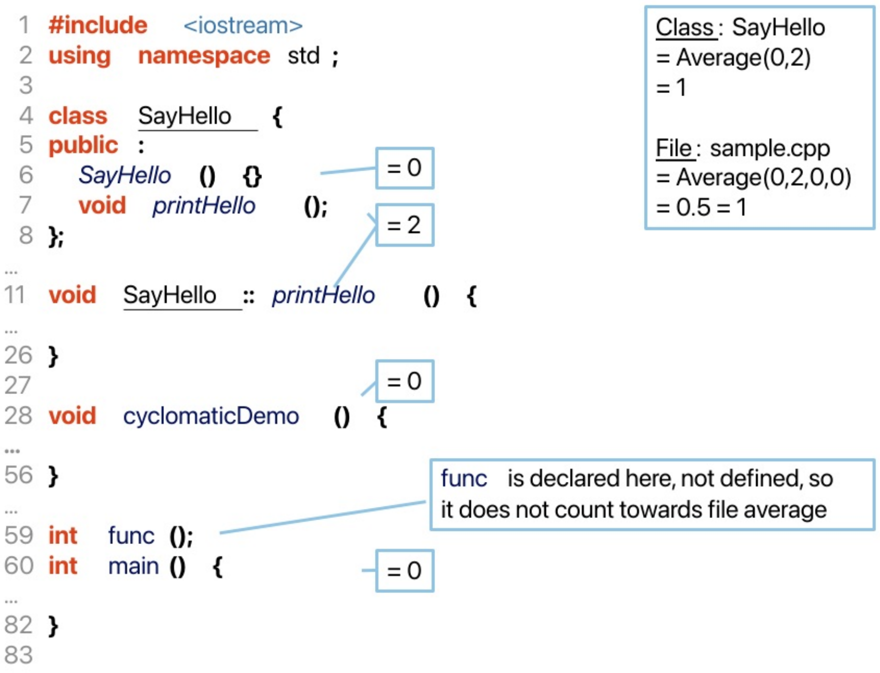
= 0

= 0

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

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

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



```
#include   <iostream>
using   namespace   std ;

class   SayHello   {
public   :
   SayHello   ()   {}
   void   printHello      ();
};
...
void   SayHello   ::  printHello      ()   {
...
}

void   cyclomaticDemo    ()   {
...
}
...
int   func  ();
int   main  ()   {
...
}

```

= 0

= 2

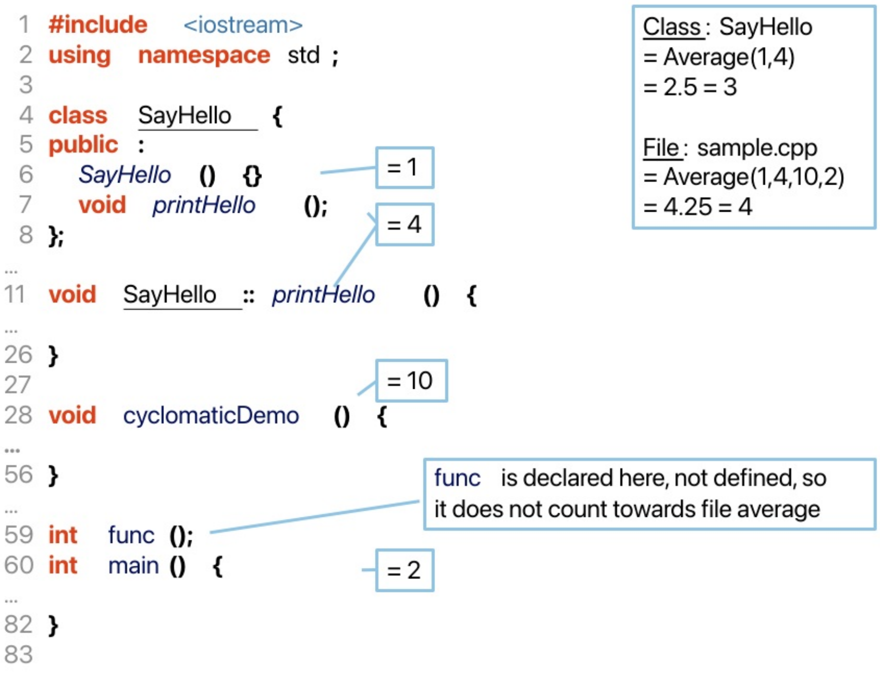
= 0

= 0

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

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

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



```
#include <iostream>
using namespace std;

class SayHello {
public:
    SayHello() {}
    void printHello();
};
...
void SayHello::printHello() {
...
}

void cyclomaticDemo() {
...
}
...
int func();
int main() {
...
}

```

= 1

= 4

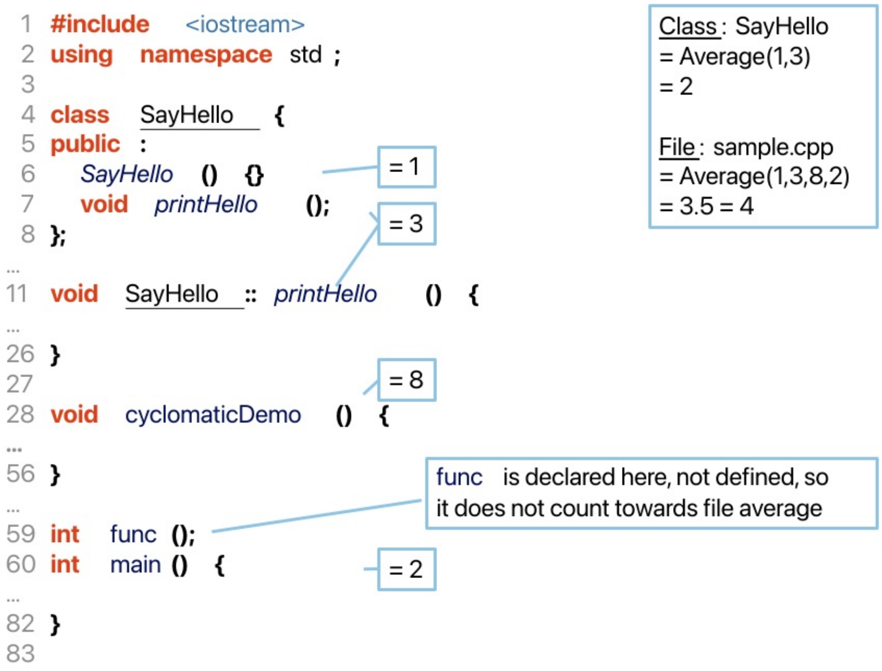
= 10

= 2

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

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

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



```
#include <iostream>
using namespace std ;

class SayHello {
public :
   SayHello () {}
   void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

= 1

= 3

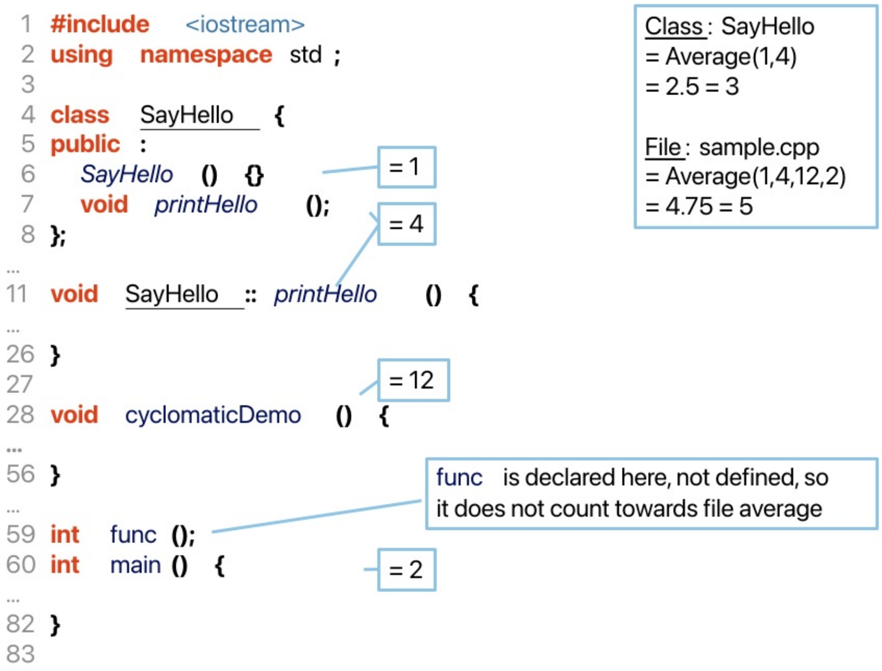
= 8

= 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

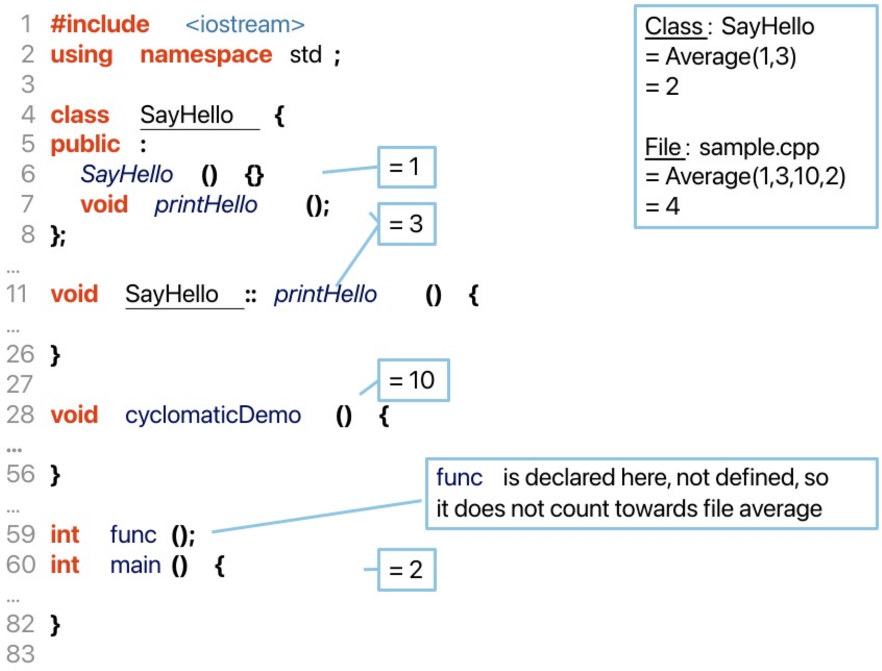


```
#include <iostream>
using namespace std ;

class SayHello {
public :
   SayHello () {}
   void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

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

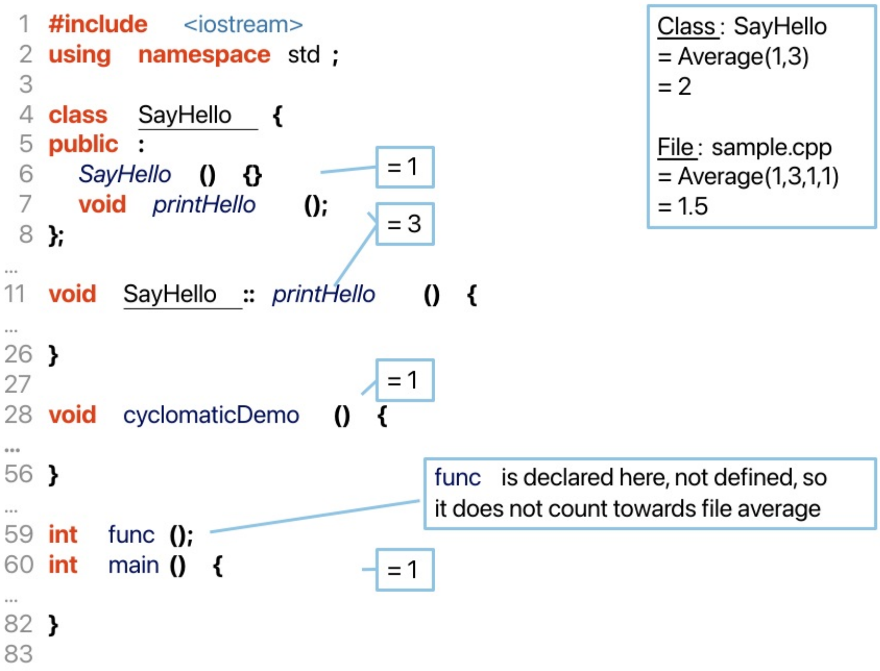


```
#include <iostream>
using namespace std ;

class SayHello {
public :
    SayHello () {}
    void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

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

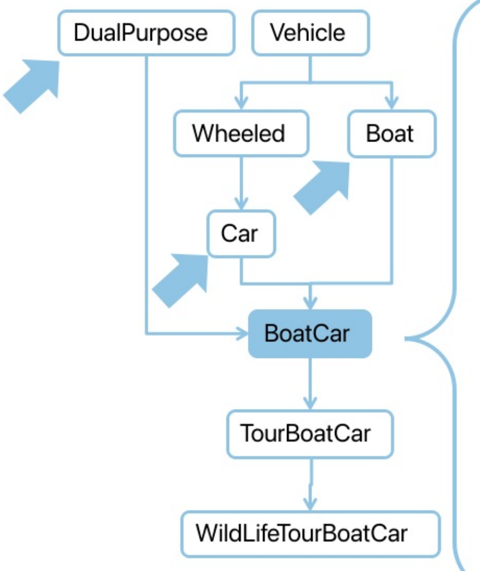


```
#include <iostream>
using namespace std ;

class SayHello {
public :
   SayHello () {}
   void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

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

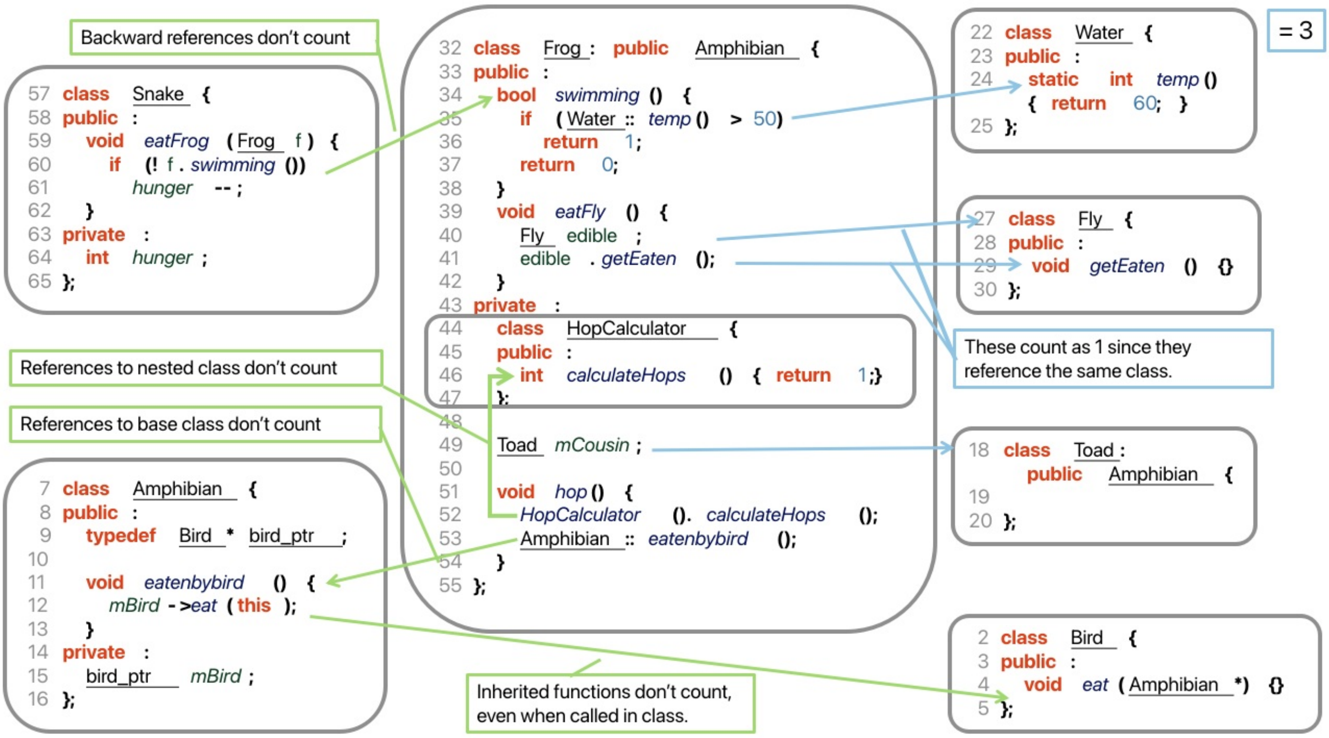
Class: SayHello
= Average(1,3)
= 2

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



```
class BoatCar : private Car, public Boat , protected DualPurpose {
public :
   // Public Instance Function
   BoatCar () : Car ( 4), Boat (), mInWater ( false ), mColor ( "Blue" ) {}
   virtual int passengers () const { return 4; }

   static int numRegistered () { return sRegistered ; }

   bool mInWater ; // Public Instance Variable

protected :

   void toggleInWater ( bool inWater ) { mInWater = inWater ; }
   char * mColor ; // Protected Instance Variable
   friend void init () {}

   static int sRegistered ;
   static double calcSpeed ( double distance , double time ) {
      return distance / time ;
   }

private :
   int mMaxPassengers ;
   void travel () {}
};
```

= 3



Backward references don't count

```
class Snake {
public :
    void eatFrog ( Frog f ) {
        if (! f . swimming ())
            hunger -- ;
    }
private :
    int hunger ;
};
```

References to nested class don't count

References to base class don't count

```
class Amphibian {
public :
    typedef Bird * bird_ptr ;

    void eatenbybird () {
        mBird ->eat ( this );
    }
private :
    bird_ptr mBird ;
};
```

```
class Frog : public Amphibian {
public :
    bool swimming () {
        if ( Water :: temp () > 50)
            return 1;
        return 0;
    }
    void eatFly () {
        Fly edible ;
        edible . getEaten ();
    }
private :
    class HopCalculator {
    public :
        int calculateHops () { return 1;}
    };

    Toad mCousin ;

    void hop () {
        HopCalculator (). calculateHops ();
        Amphibian :: eatenbybird ();
    }
};
```

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

```
class Water {
public :
    static int temp ()
    { return 60; }
};
```

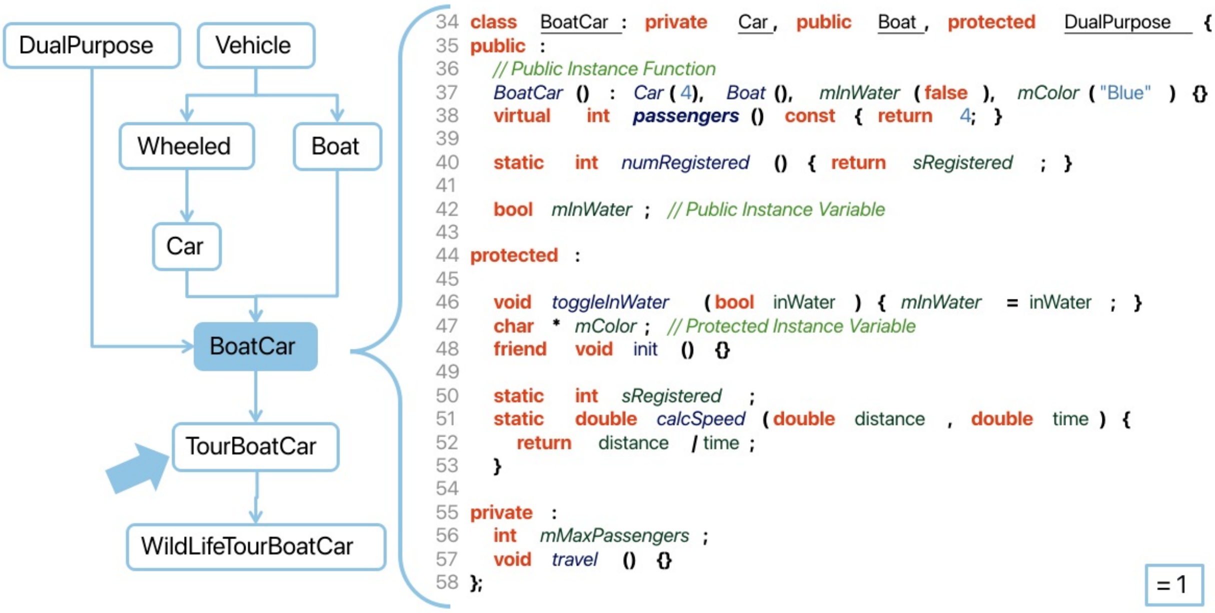
= 3

```
class Fly {
public :
    void getEaten () {}
};
```

These count as 1 since they reference the same class.

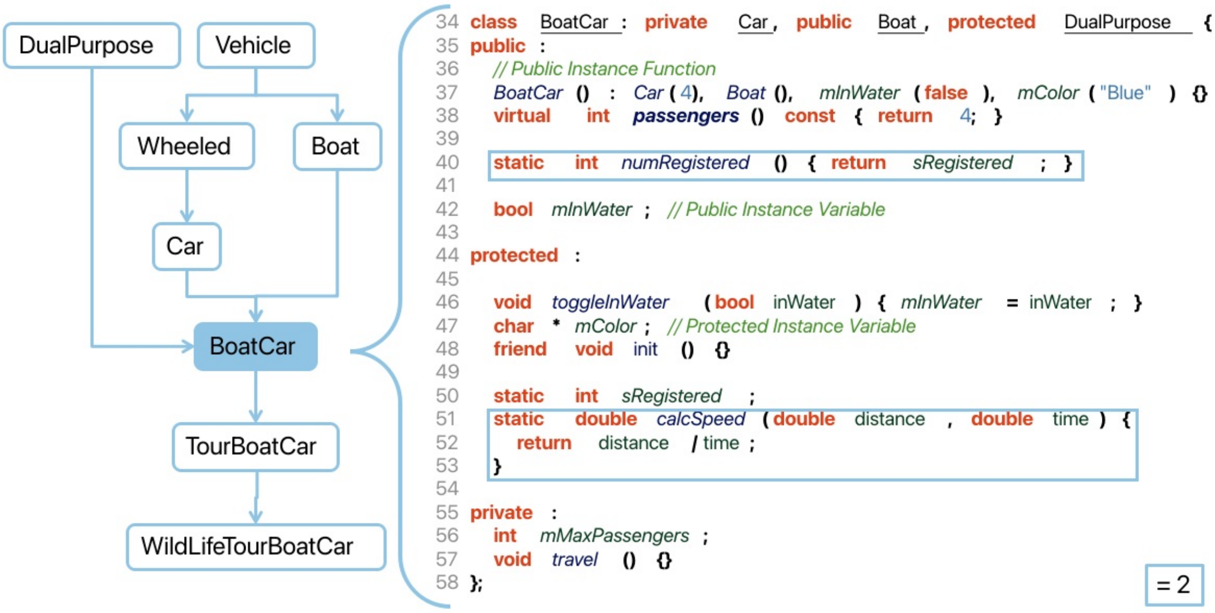
```
class Toad :
    public Amphibian {

};
```

```
class Bird {
public :
    void eat ( Amphibian *) {}
};
```



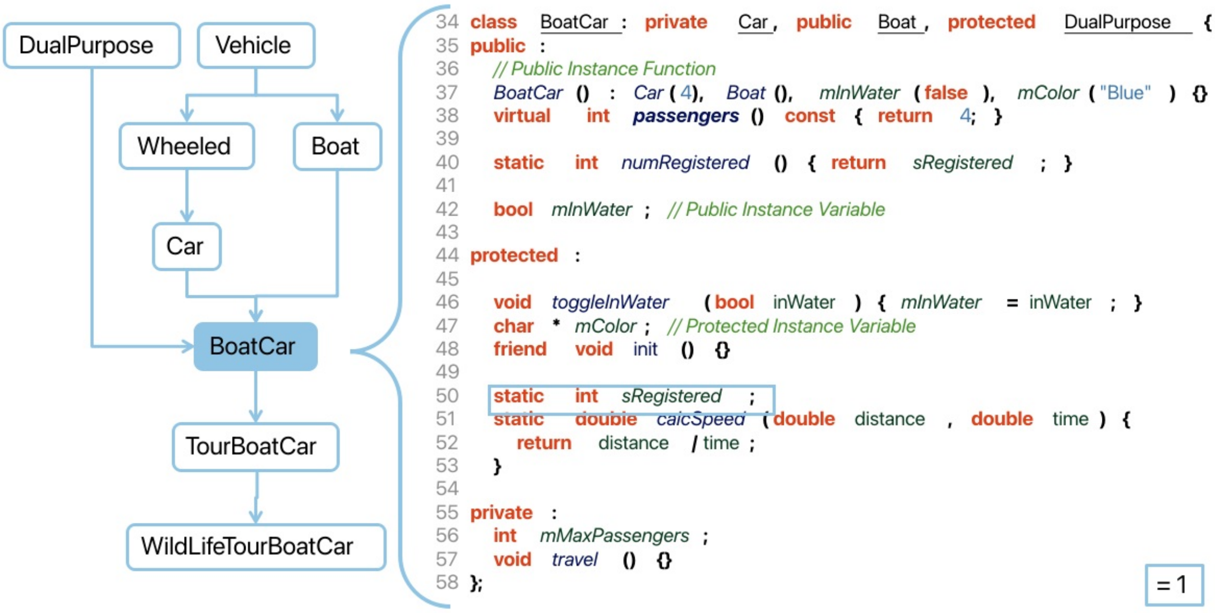
```
class BoatCar : private Car, public Boat, protected DualPurpose {
public :
   // Public Instance Function
   BoatCar () : Car (4), Boat (), mInWater (false ), mColor ("Blue" ) {}
   virtual int passengers () const { return 4; }

   static int numRegistered () { return sRegistered ; }

   bool mInWater ; // Public Instance Variable

protected :

   void toggleInWater (bool inWater ) { mInWater = inWater ; }
   char * mColor ; // Protected Instance Variable
   friend void init () {}

   static int sRegistered ;
   static double calcSpeed (double distance , double time ) {
      return distance / time ;
   }

private :
   int mMaxPassengers ;
   void travel () {}
};
```

= 1

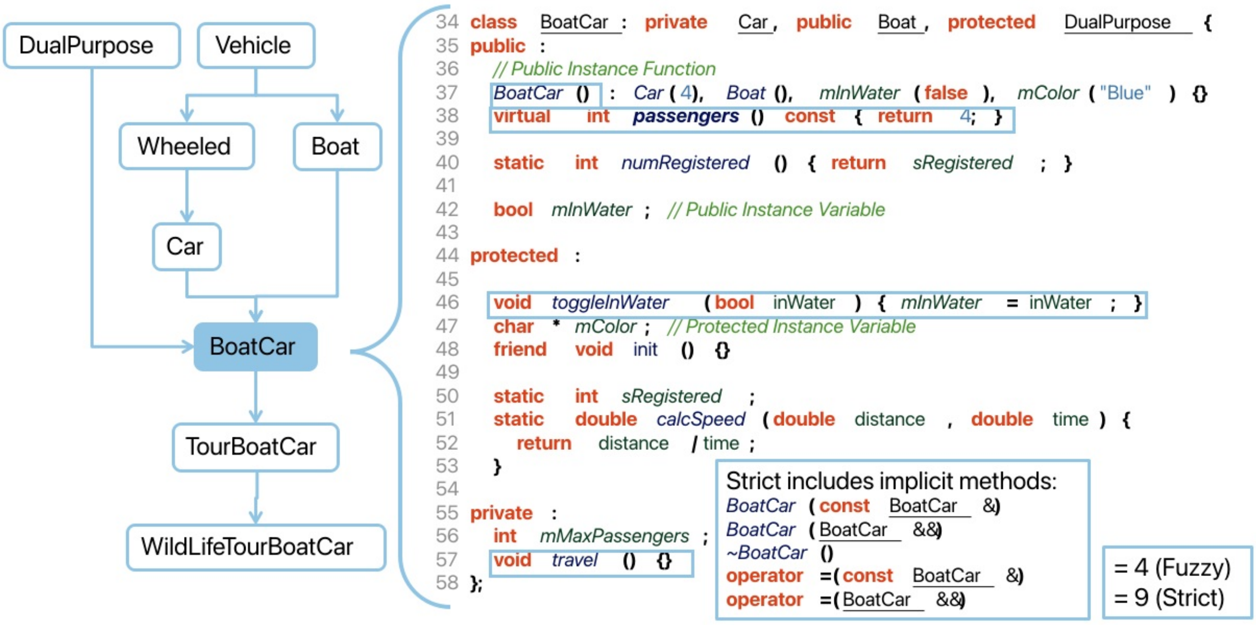


```
class BoatCar : private Car, public Boat, protected DualPurpose {
public :
  // Public Instance Function
  BoatCar () : Car ( 4), Boat (), mInWater ( false ), mColor ( "Blue" ) {}
  virtual int passengers () const { return 4; }

  static int numRegistered () { return sRegistered ; }

  bool mInWater ; // Public Instance Variable

protected :

  void toggleInWater ( bool inWater ) { mInWater = inWater ; }
  char * mColor ; // Protected Instance Variable
  friend void init () {}

  static int sRegistered ;
  static double calcSpeed ( double distance , double time ) {
    return distance / time ;
  }

private :
  int mMaxPassengers ;
  void travel () {}
};
```

= 2



```
class BoatCar : private Car , public Boat , protected DualPurpose {
public :
  // Public Instance Function
  BoatCar () : Car ( 4), Boat (), mInWater ( false ), mColor ( "Blue" ) {}
  virtual int passengers () const { return 4; }

  static int numRegistered () { return sRegistered ; }

  bool mInWater ; // Public Instance Variable

protected :

  void toggleInWater ( bool inWater ) { mInWater = inWater ; }
  char * mColor ; // Protected Instance Variable
  friend void init () {}

  static int sRegistered ;
  static double calcSpeed ( double distance , double time ) {
    return distance / time ;
  }

private :
  int mMaxPassengers ;
  void travel () {}
};
```

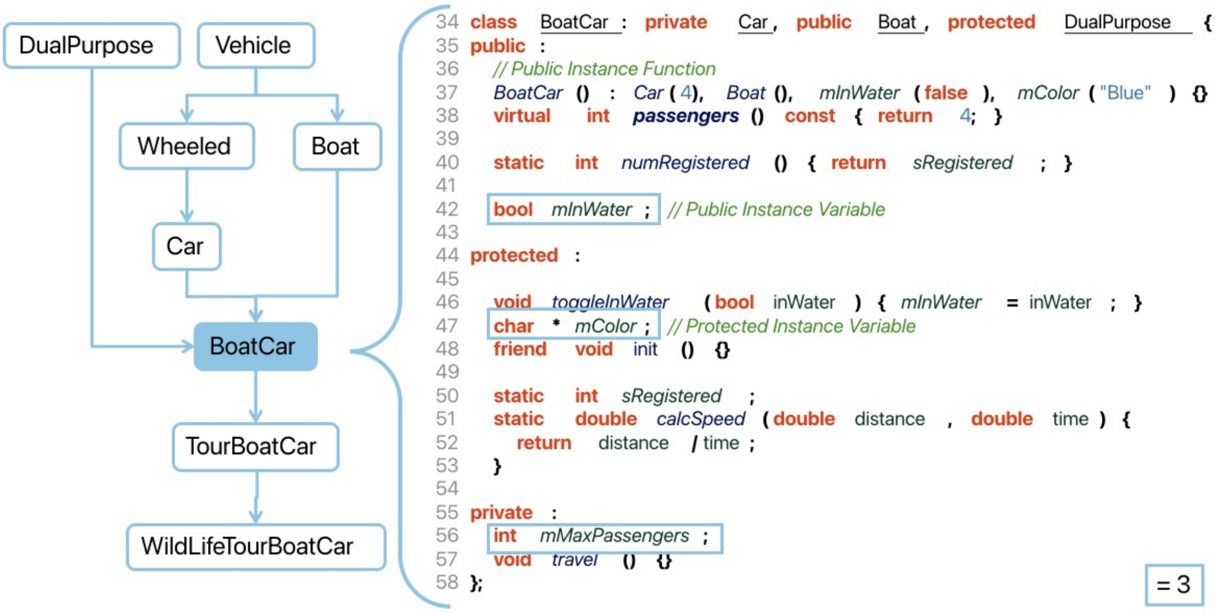


```
class BoatCar : private Car , public Boat , protected DualPurpose {
public :
   // Public Instance Function
   BoatCar () : Car ( 4), Boat (), mInWater ( false ), mColor ( "Blue" ) {}
   virtual int passengers () const { return 4; }

   static int numRegistered () { return sRegistered ; }

   bool mInWater ; // Public Instance Variable

protected :

   void toggleInWater ( bool inWater ) { mInWater = inWater ; }
   char * mColor ; // Protected Instance Variable
   friend void init () {}

   static int sRegistered ;
   static double calcSpeed ( double distance , double time ) {
      return distance / time ;
   }

private :
   int mMaxPassengers ;
   void travel () {}
};
```

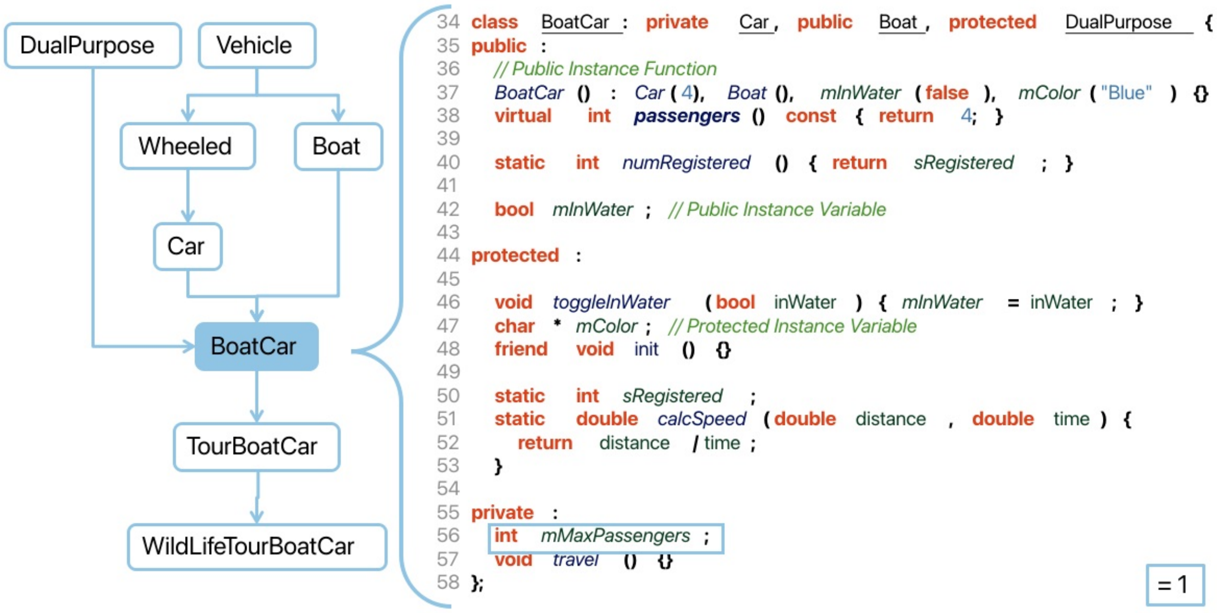
Strict includes implicit methods:

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

= 4 (Fuzzy)
= 9 (Strict)

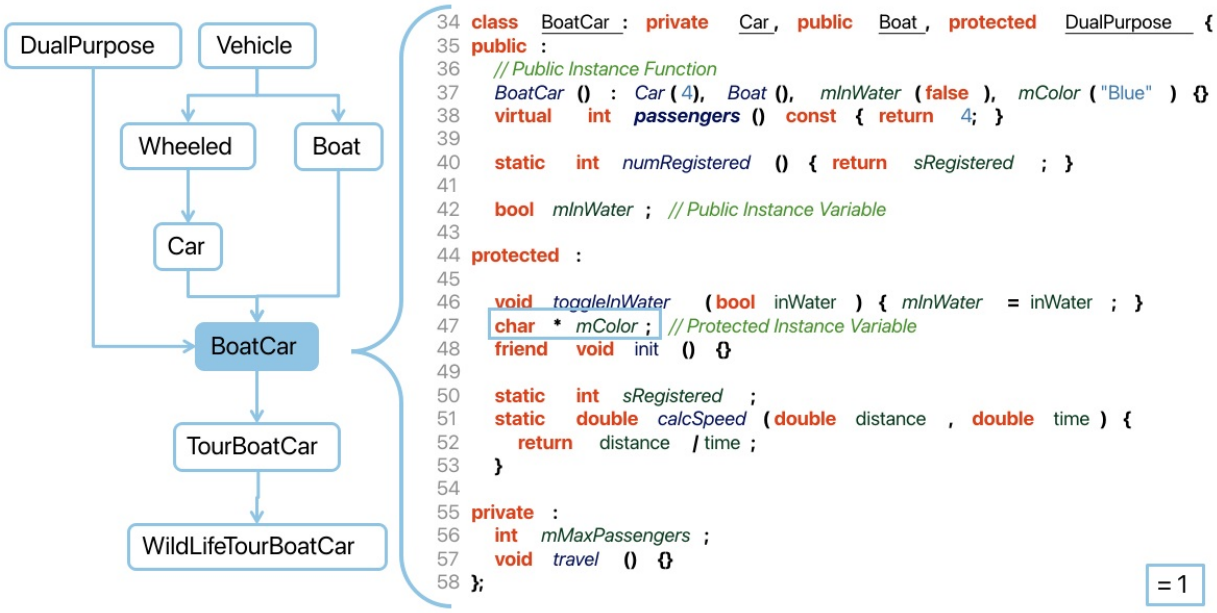


```
class BoatCar : private Car , public Boat , protected DualPurpose {
public :
  // Public Instance Function
  BoatCar () : Car ( 4), Boat (), mInWater ( false ), mColor ( "Blue" ) {}
  virtual int passengers () const { return 4; }

  static int numRegistered () { return sRegistered ; }

  bool mInWater ; // Public Instance Variable

protected :

  void toggleInWater ( bool inWater ) { mInWater = inWater ; }
  char * mColor ; // Protected Instance Variable
  friend void init () {}

  static int sRegistered ;
  static double calcSpeed ( double distance , double time ) {
    return distance / time ;
  }

private :
  int mMaxPassengers ;
  void travel () {}
};
```



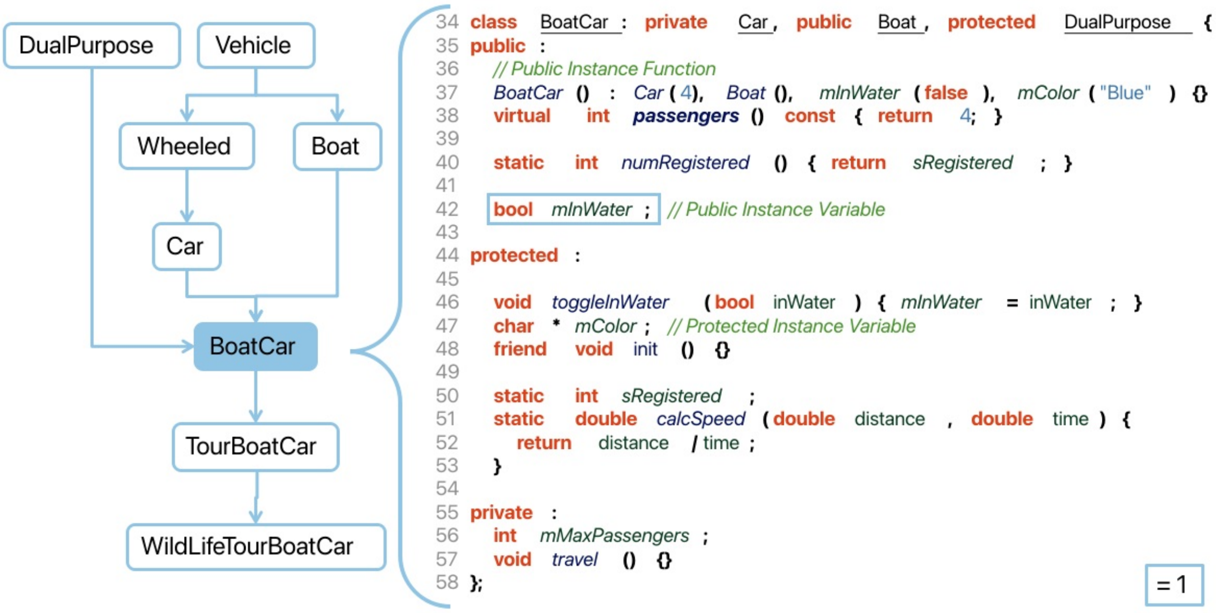
```
class BoatCar : private Car, public Boat, protected DualPurpose {
public :
   // Public Instance Function
   BoatCar () : Car ( 4), Boat (), mInWater ( false ), mColor ( "Blue" ) {}
   virtual int passengers () const { return 4; }

   static int numRegistered () { return sRegistered ; }

   bool mInWater ; // Public Instance Variable

protected :

   void toggleInWater ( bool inWater ) { mInWater = inWater ; }
   char * mColor ; // Protected Instance Variable
   friend void init () {}

   static int sRegistered ;
   static double calcSpeed ( double distance , double time ) {
      return distance / time ;
   }

private :
   int mMaxPassengers ;
   void travel () {}
};
```

= 1

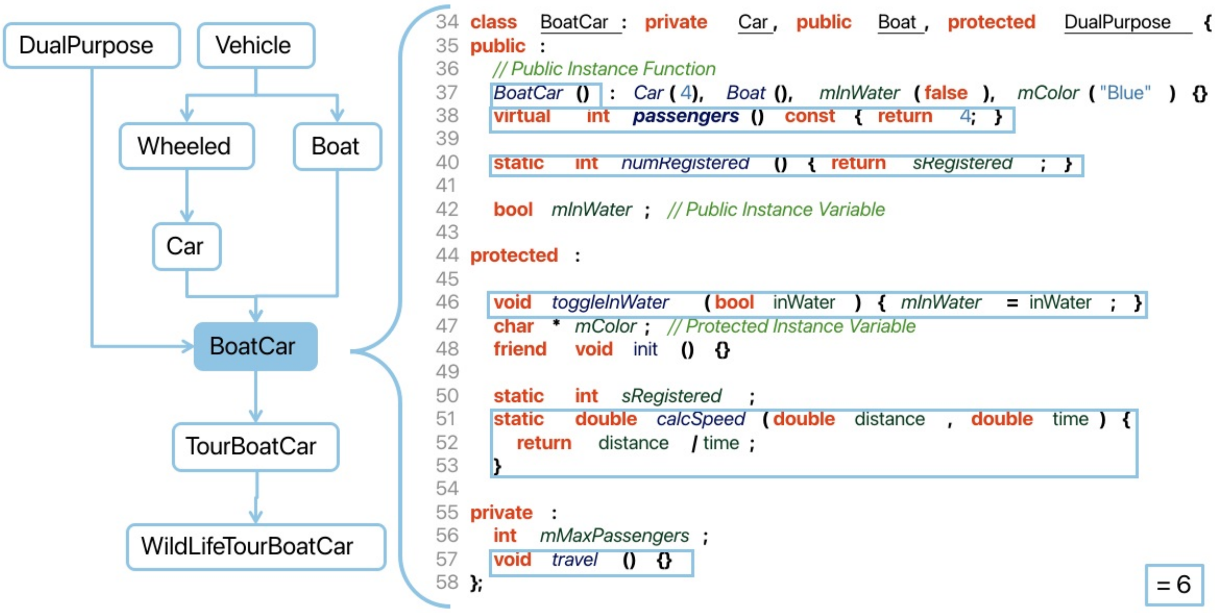


```
class BoatCar : private Car, public Boat, protected DualPurpose {
public :
   // Public Instance Function
   BoatCar () : Car ( 4), Boat (), mInWater ( false ), mColor ( "Blue" ) {}
   virtual int passengers () const { return 4; }

   static int numRegistered () { return sRegistered ; }

   bool mInWater ; // Public Instance Variable

protected :

   void toggleInWater ( bool inWater ) { mInWater = inWater ; }
   char * mColor ; // Protected Instance Variable
   friend void init () {}

   static int sRegistered ;
   static double calcSpeed ( double distance , double time ) {
      return distance / time ;
   }

private :
   int mMaxPassengers ;
   void travel () {}
};
```

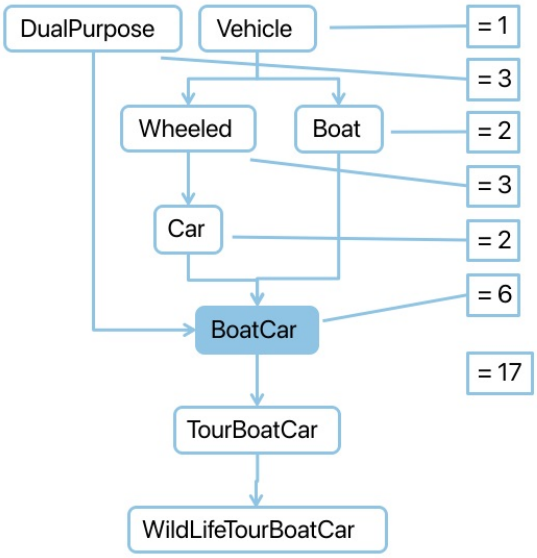
= 1

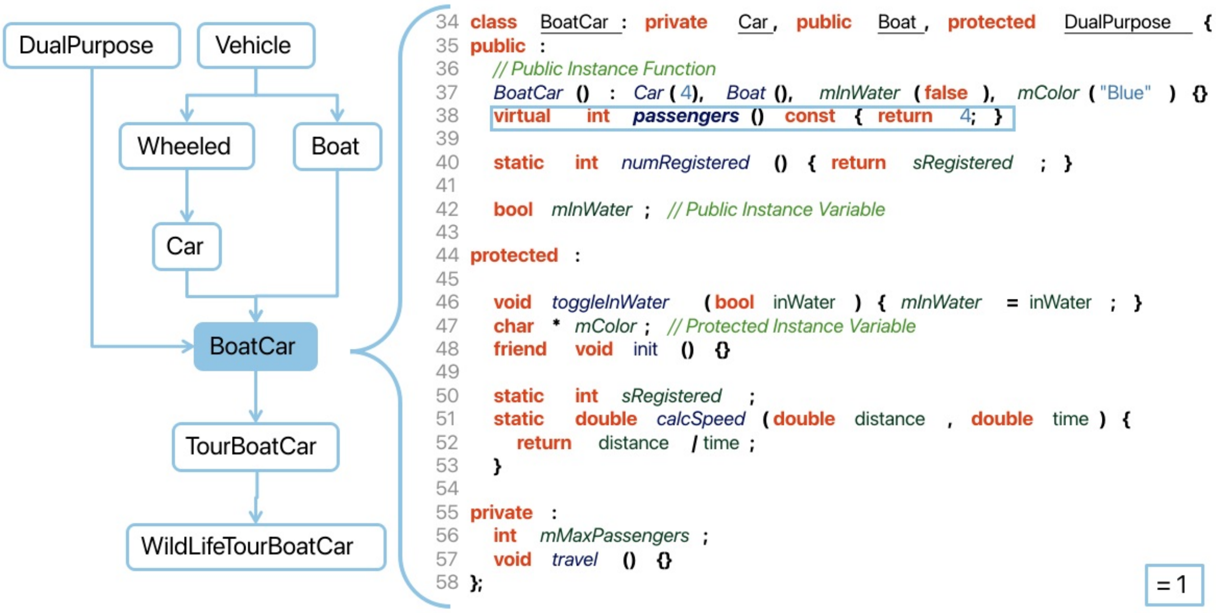


```
class BoatCar : private Car, public Boat, protected DualPurpose {
public :
   // Public Instance Function
   BoatCar () : Car ( 4), Boat (), mInWater ( false ), mColor ( "Blue" ) {}
   virtual int passengers () const { return 4; }

   static int numRegistered () { return sRegistered ; }

   bool mInWater ; // Public Instance Variable

protected :

   void toggleInWater ( bool inWater ) { mInWater = inWater ; }
   char * mColor ; // Protected Instance Variable
   friend void init () {}

   static int sRegistered ;
   static double calcSpeed ( double distance , double time ) {
      return distance / time ;
   }

private :
   int mMaxPassengers ;
   void travel () {}
};
```

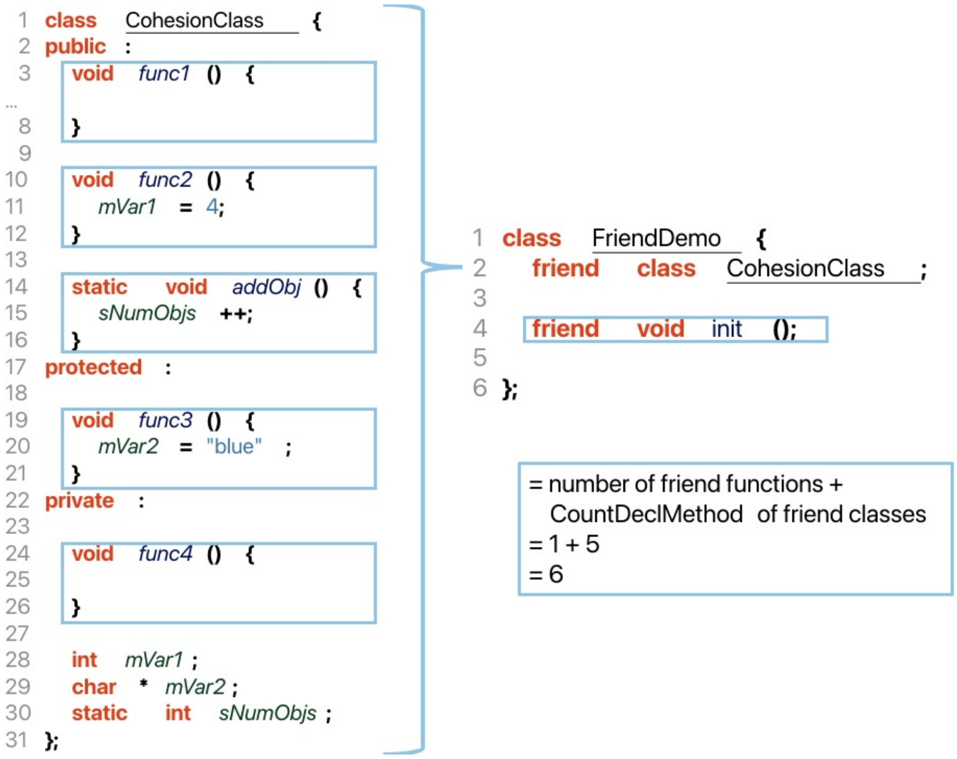


```
class BoatCar : private Car, public Boat, protected DualPurpose {
public :
  // Public Instance Function
  BoatCar () : Car (4), Boat (), mInWater (false ), mColor ("Blue" ) {}
  virtual int passengers () const { return 4; }

  static int numRegistered () { return sRegistered ; }

  bool mInWater ; // Public Instance Variable

protected :

  void toggleInWater (bool inWater ) { mInWater = inWater ; }
  char * mColor ; // Protected Instance Variable
  friend void init () {}

  static int sRegistered ;
  static double calcSpeed (double distance , double time ) {
    return distance / time ;
  }

private :
  int mMaxPassengers ;
  void travel () {}
};
```





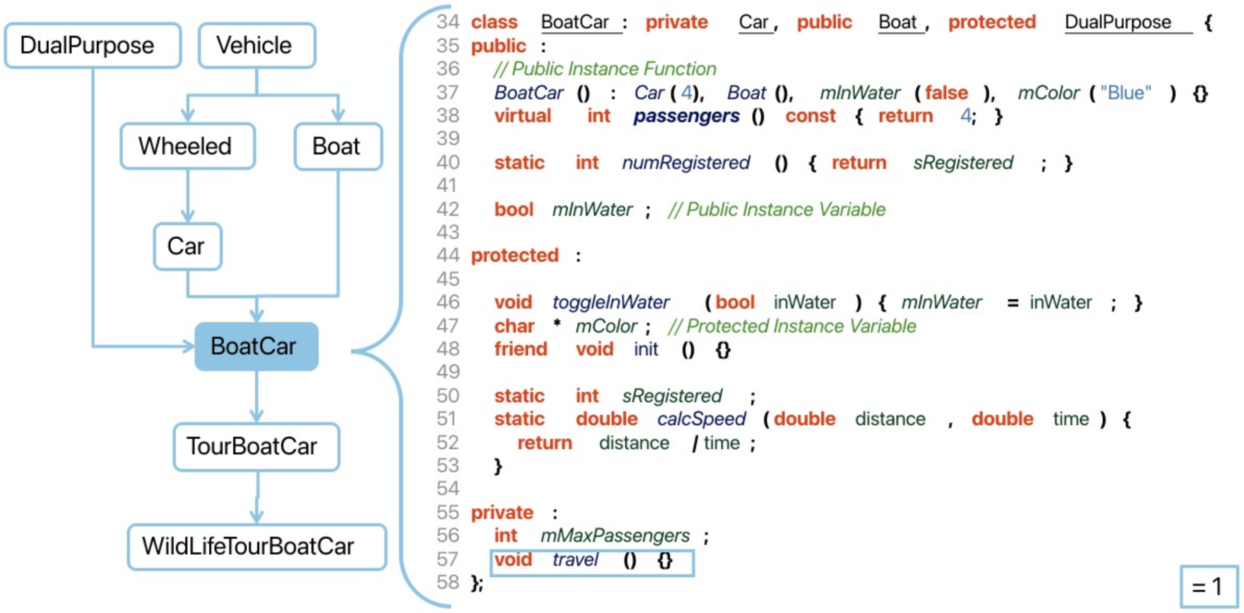
```
class BoatCar : private Car, public Boat, protected DualPurpose {
public :
   // Public Instance Function
   BoatCar () : Car (4), Boat (), mInWater (false), mColor ("Blue") {}
   virtual int passengers () const { return 4; }

   static int numRegistered () { return sRegistered ; }

   bool mInWater ; // Public Instance Variable

protected :

   void toggleInWater (bool inWater) { mInWater = inWater ; }
   char * mColor ; // Protected Instance Variable
   friend void init () {}

   static int sRegistered ;
   static double calcSpeed (double distance , double time ) {
      return distance / time ;
   }

private :
   int mMaxPassengers ;
   void travel () {}
};
```



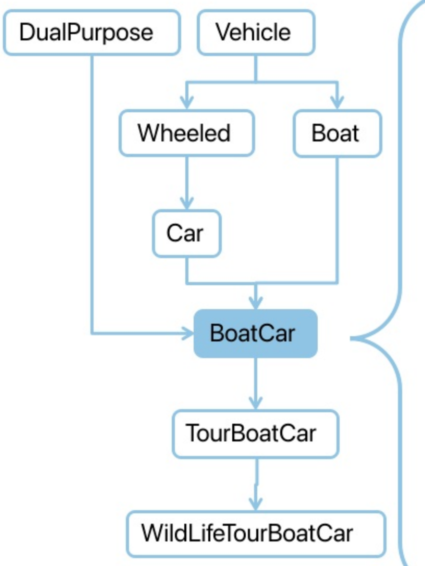
```
class CohesionClass {
public :
   void func1 () {
...
   }

   void func2 () {
      mVar1 = 4;
   }

   static void addObj () {
      sNumObjs ++;
   }
protected :

   void func3 () {
      mVar2 = "blue" ;
   }
private :

   void func4 () {

   }

   int mVar1 ;
   char * mVar2 ;
   static int sNumObjs ;
};
```

```
class FriendDemo {
   friend class CohesionClass ;

   friend void init ();

};
```

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

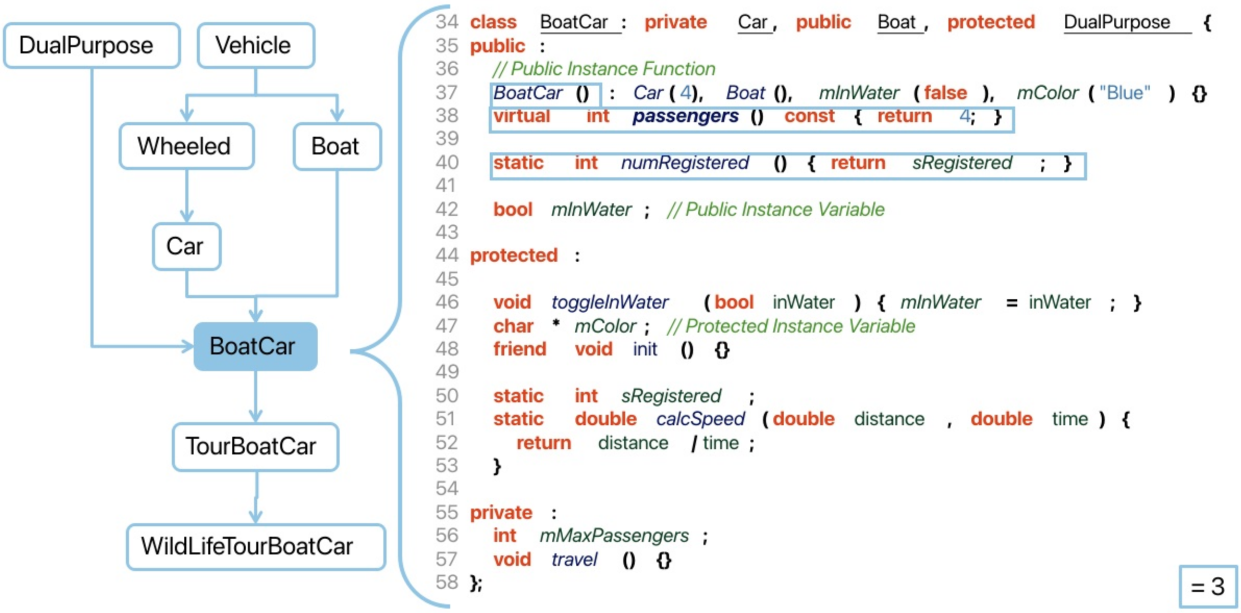


```
class BoatCar : private Car, public Boat, protected DualPurpose {
public :
    // Public Instance Function
    BoatCar () : Car (4), Boat (), mInWater (false ), mColor ("Blue" ) {}
    virtual int passengers () const { return 4; }

    static int numRegistered () { return sRegistered ; }

    bool mInWater ; // Public Instance Variable

protected :

    void toggleInWater (bool inWater ) { mInWater = inWater ; }
    char * mColor ; // Protected Instance Variable
    friend void init () {}

    static int sRegistered ;
    static double calcSpeed (double distance , double time ) {
      return distance / time ;
    }

private :
    int mMaxPassengers ;
    void travel () {}
};
```



```
class BoatCar : private Car, public Boat, protected DualPurpose {
public :
  // Public Instance Function
  BoatCar () : Car (4), Boat (), mInWater (false ), mColor ("Blue" ) {}
  virtual int passengers () const { return 4; }

  static int numRegistered () { return sRegistered ; }

  bool mInWater ; // Public Instance Variable

protected :

  void toggleInWater (bool inWater ) { mInWater = inWater ; }
  char * mColor ; // Protected Instance Variable
  friend void init () {}

  static int sRegistered ;
  static double calcSpeed (double distance , double time ) {
    return distance / time ;
  }

private :
  int mMaxPassengers ;
  void travel () {}
};
```

= 2

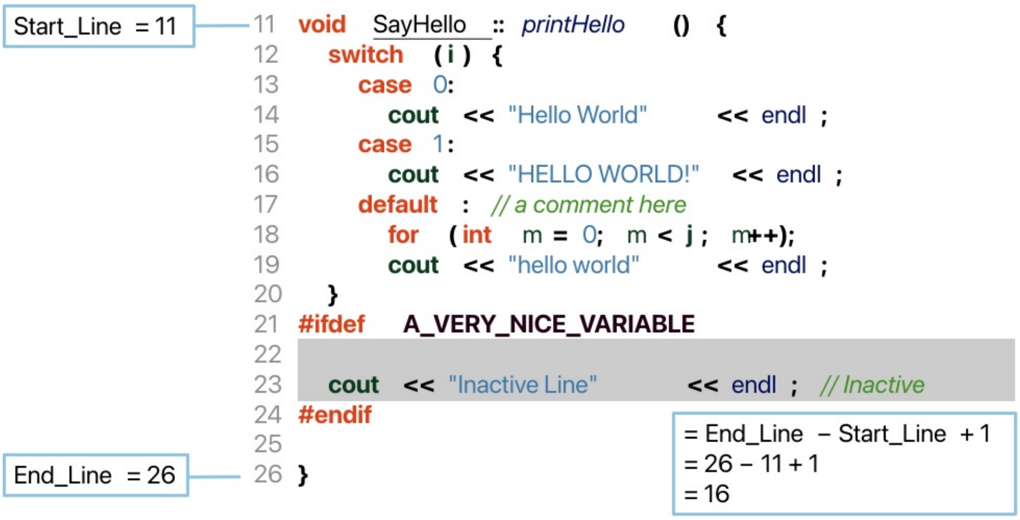


```
class BoatCar : private Car, public Boat, protected DualPurpose {
public :
  // Public Instance Function
  BoatCar () : Car ( 4), Boat (), mInWater ( false ), mColor ( "Blue" ) {}
  virtual int passengers () const { return 4; }

  static int numRegistered () { return sRegistered ; }

  bool mInWater ; // Public Instance Variable

protected :

  void toggleInWater ( bool inWater ) { mInWater = inWater ; }
  char * mColor ; // Protected Instance Variable
  friend void init () {}

  static int sRegistered ;
  static double calcSpeed ( double distance , double time ) {
    return distance / time ;
  }

private :
  int mMaxPassengers ;
  void travel () {}
};
```

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

```
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);
```

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

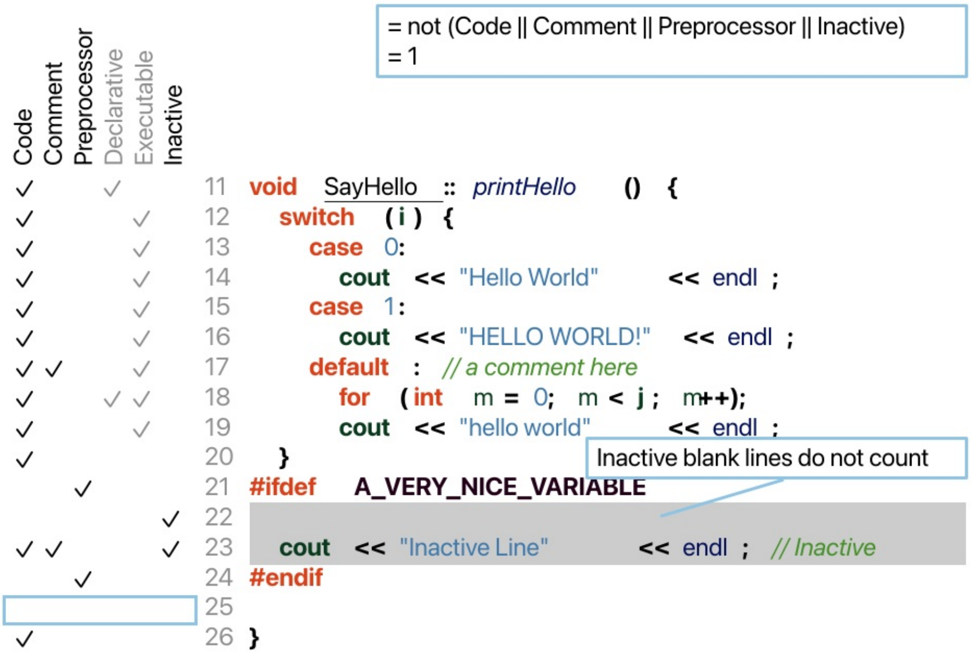


```
11 void SayHello :: printHello () {
12   switch ( i ) {
13     case 0:
14       cout << "Hello World" << endl ;
15     case 1:
16       cout << "HELLO WORLD!" << endl ;
17     default : // a comment here
18       for ( int m = 0; m < j ; m++);
19       cout << "hello world" << endl ;
20   }
21 #ifdef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

Start_Line = 11

End_Line = 26

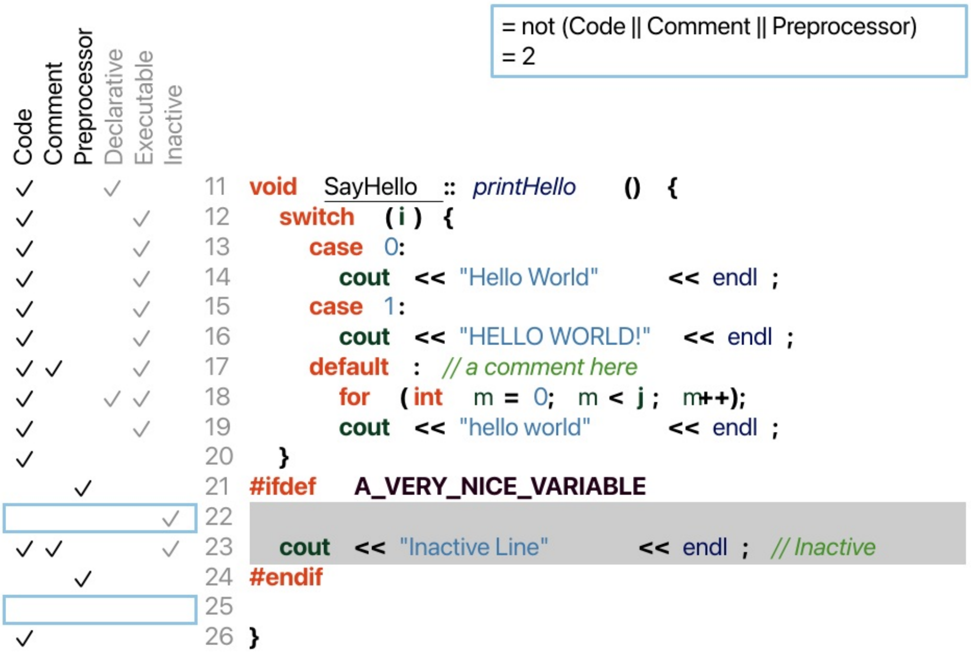
= End_Line − Start_Line + 1
= 26 − 11 + 1
= 16



= not (Code || Comment || Preprocessor || Inactive)
= 1

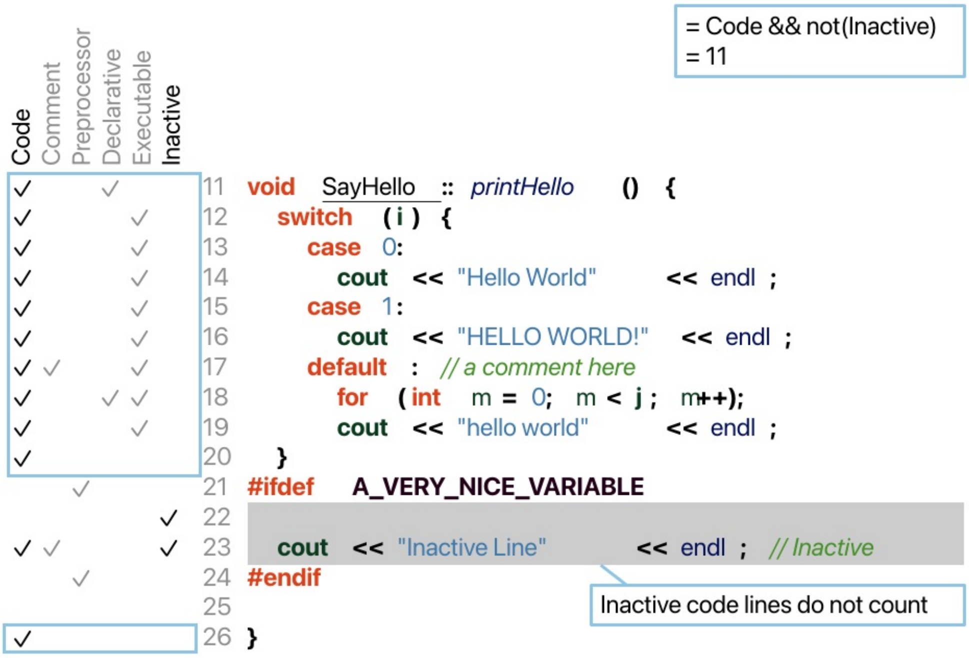
Code	Comment	Preprocessor	Declarative	Executable	Inactive		
✓			✓			11	`void SayHello :: printHello () {`
✓				✓		12	`switch (i) {`
✓				✓		13	`case 0:`
✓				✓		14	`cout << "Hello World" << endl ;`
✓				✓		15	`case 1:`
✓				✓		16	`cout << "HELLO WORLD!" << endl ;`
✓	✓			✓		17	`default : // a comment 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	`}`

Inactive blank lines do not count



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

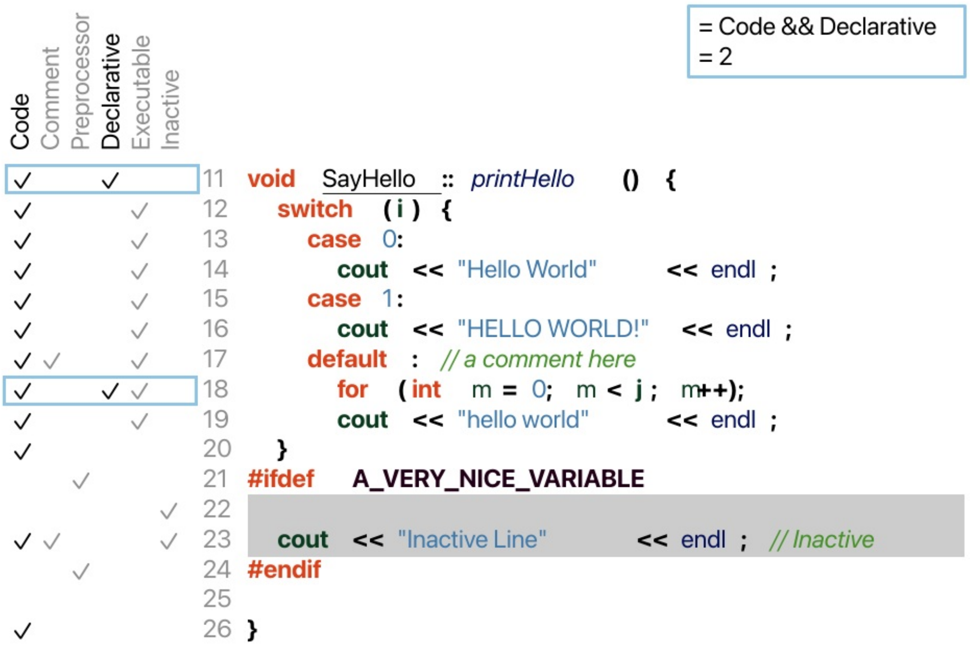
Code	Comment	Preprocessor	Declarative	Executable	Inactive		
✓			✓			11	`void SayHello :: printHello () {`
✓				✓		12	`switch ( i ) {`
✓				✓		13	`case 0:`
✓				✓		14	`cout << "Hello World" << endl ;`
✓				✓		15	`case 1 :`
✓				✓		16	`cout << "HELLO WORLD!" << endl ;`
✓	✓			✓		17	`default : // a comment 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	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	`}`

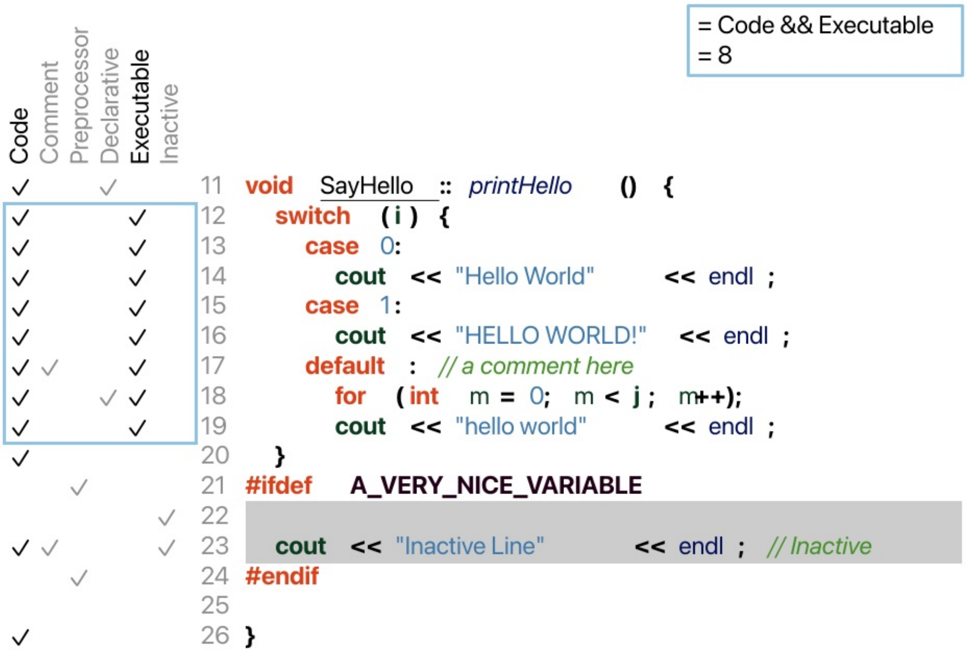
= Code && not(Inactive)
= 11

Inactive code lines do not count



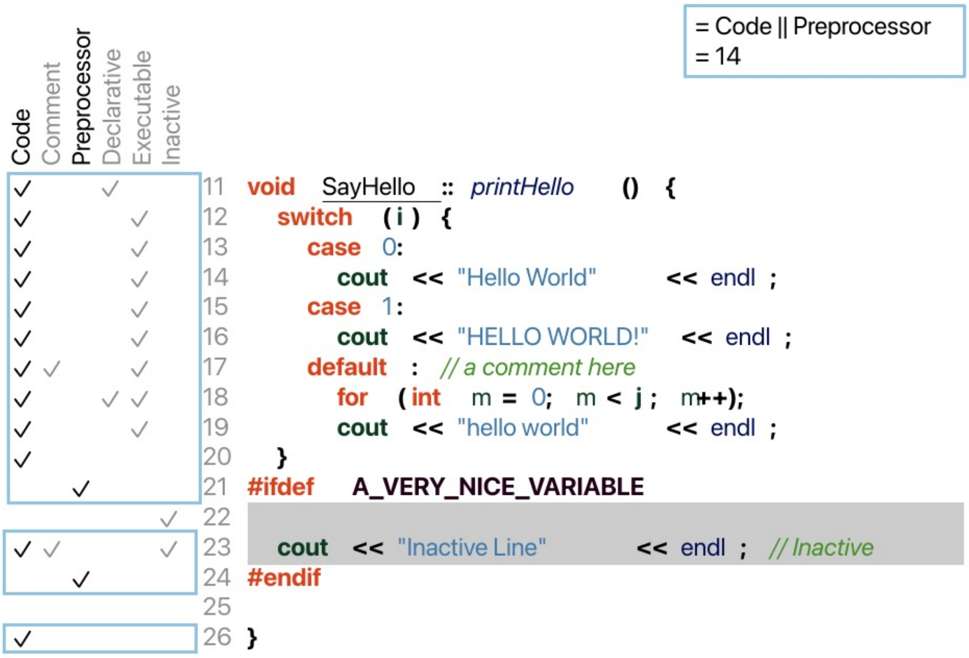
= Code && Declarative
= 2

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	}



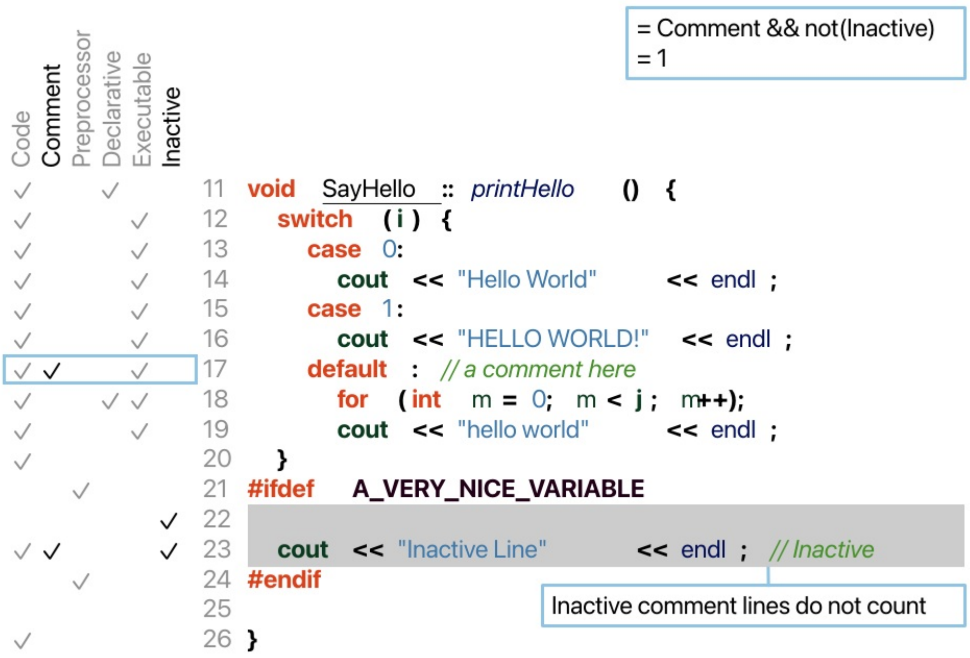
= Code && Executable
= 8

Code	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 || Preprocessor
= 14

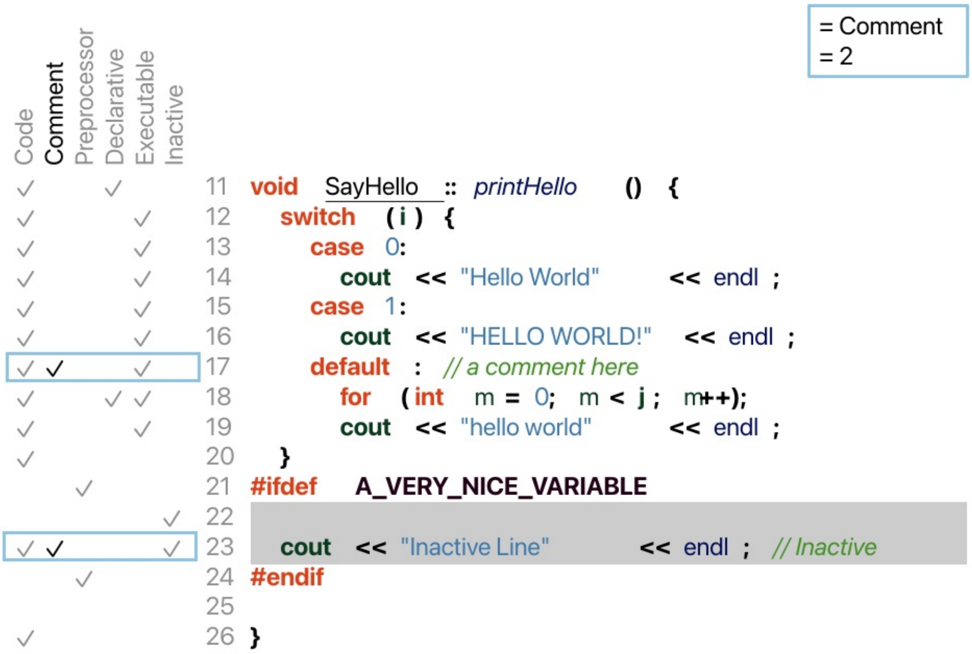
Code	Comment	Preprocessor	Declarative	Executable	Inactive		
✓			✓			11	`void SayHello :: printHello () {`
✓				✓		12	`switch (i) {`
✓				✓		13	`case 0:`
✓				✓		14	`cout << "Hello World" << endl ;`
✓				✓		15	`case 1:`
✓				✓		16	`cout << "HELLO WORLD!" << endl ;`
✓	✓			✓		17	`default : // a comment 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 ; // Inactive`
		✓				24	`#endif`
						25	
✓						26	`}`



= Comment && not(Inactive)
= 1

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	`}`

Inactive comment lines do not count

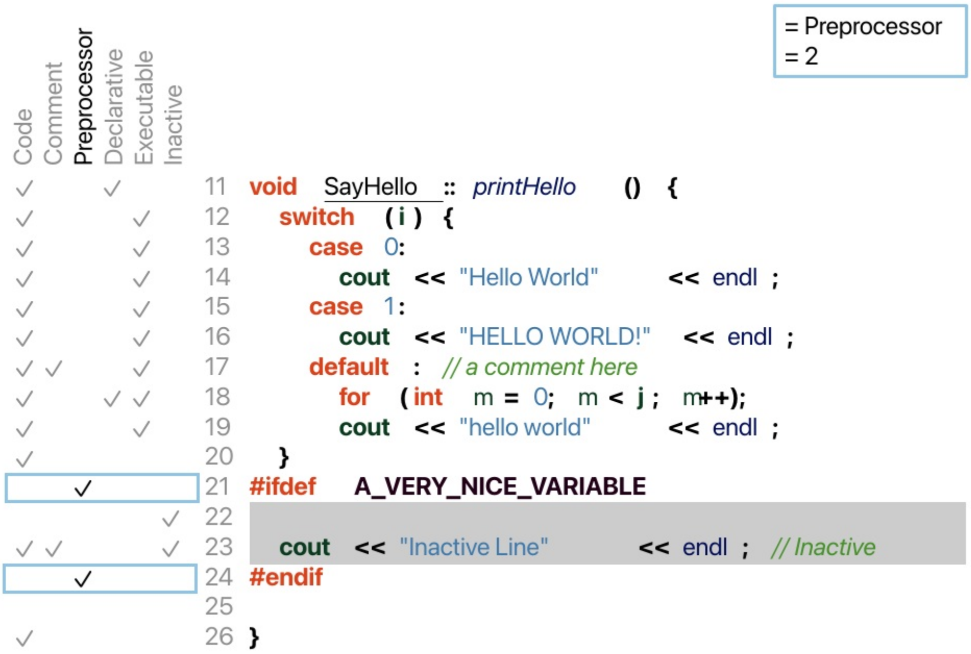


= Comment
= 2

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	`}`

= Inactive
= 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	`}`



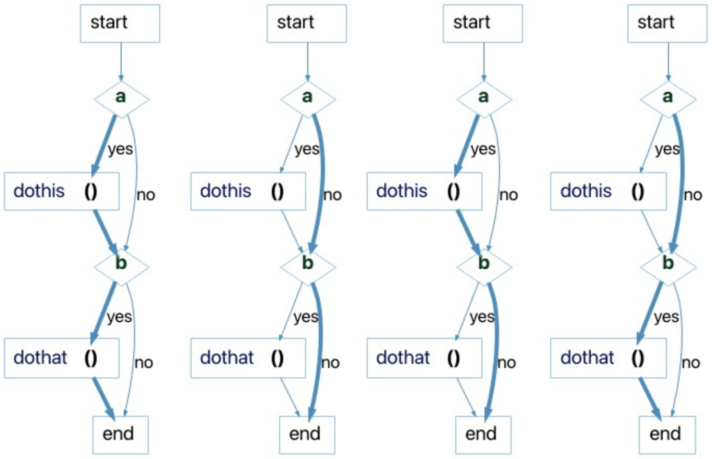
= Preprocessor
= 2

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	}

```
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 }
```

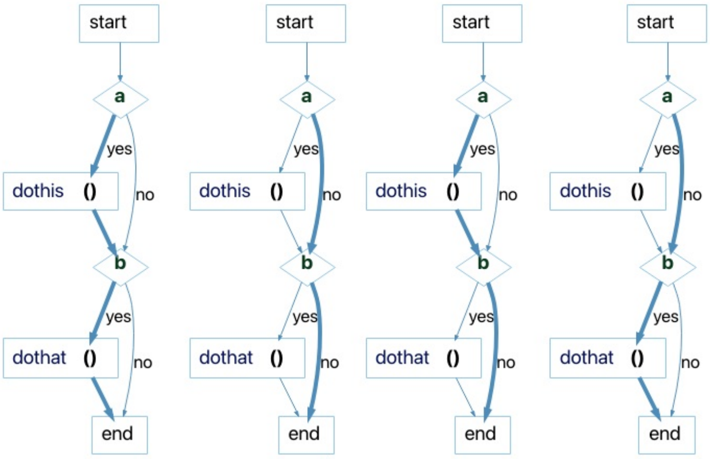
= functions called + parameters set + globals set + non -void return type
= 1 + 4 + 1 + 1
= 7

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



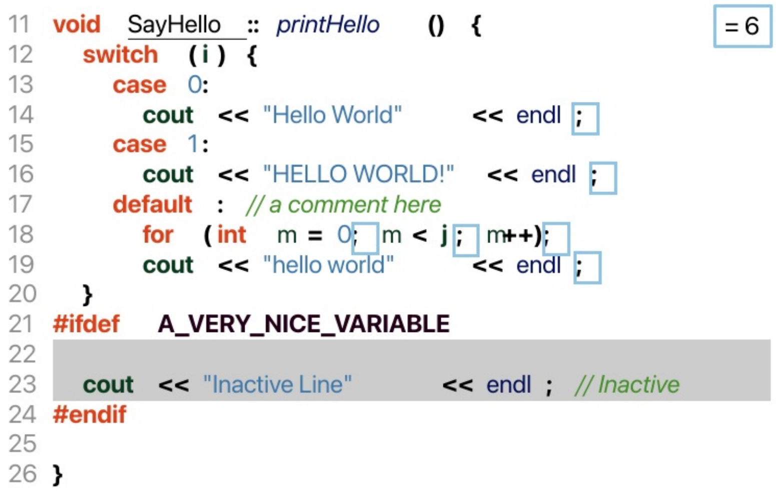
= 4

```
void pathDemo () {
    if ( a)
        dothis ();
    if ( b)
        dothat ();
}
```

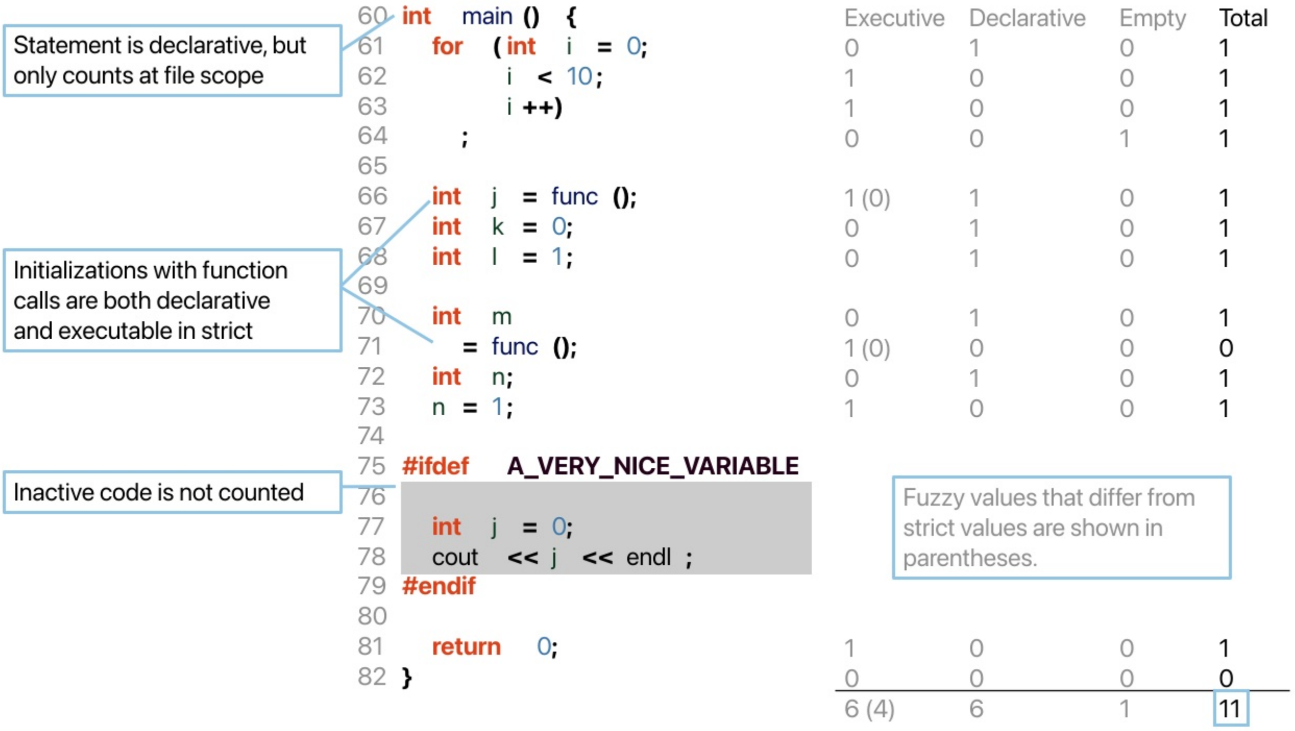


$= \mathrm{Log}_{10}(4)$
$= 1$

```
void  pathDemo ()  {
   if   ( a)
        dothis   ();
   if   ( b)
        dothat   ();
}
```



```
void  SayHello :: printHello  () {
  switch  ( i ) {
    case  0:
      cout  << "Hello World"    << endl ;
    case  1:
      cout  << "HELLO WORLD!"   << endl ;
    default  : // a comment here
      for  ( int  m = 0; m < j ; m++);
      cout  << "hello world"    << endl ;
  }
#ifdef  A_VERY_NICE_VARIABLE

  cout  << "Inactive Line"    << endl ; // Inactive
#endif

}
```



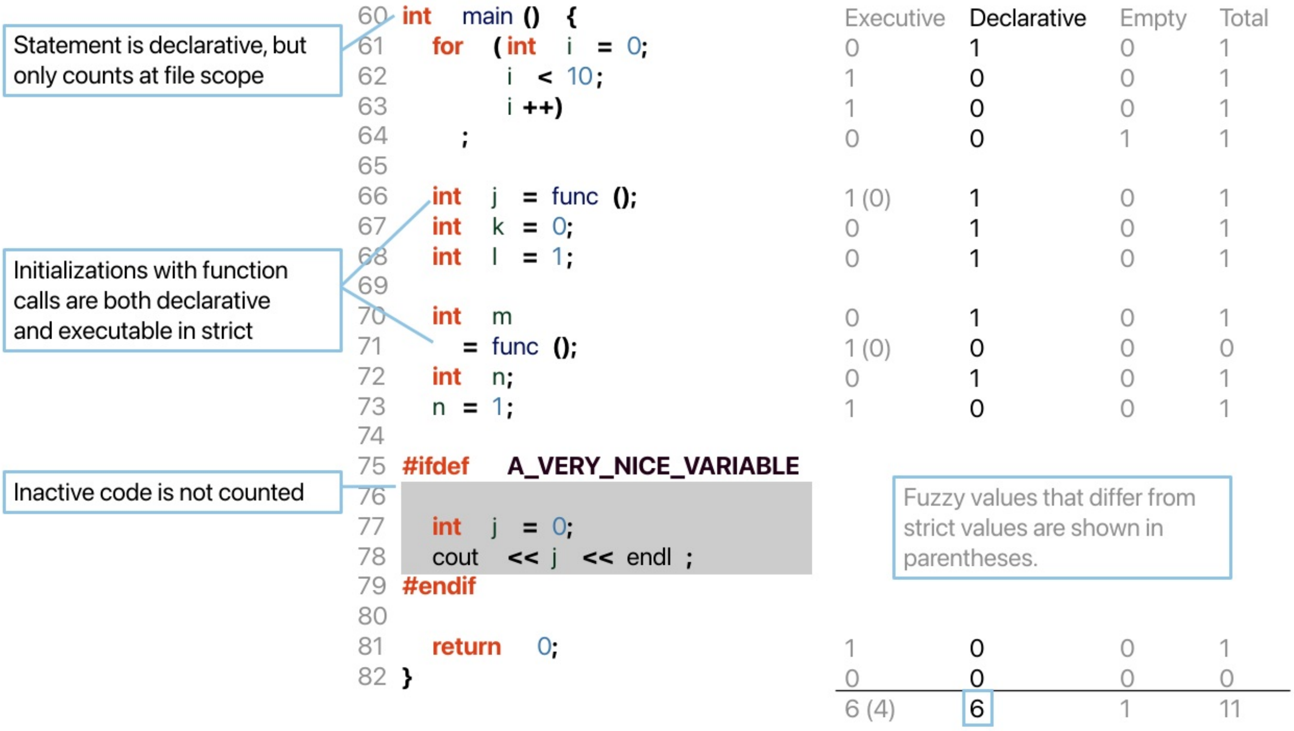
```
int main () {                                  Executive  Declarative  Empty  Total
    for ( int i = 0;                           0          1            0      1
          i < 10;                              1          0            0      1
          i ++)                                1          0            0      1
      ;                                        0          0            1      1

    int j = func ();                           1 (0)      1            0      1
    int k = 0;                                 0          1            0      1
    int l = 1;                                 0          1            0      1

    int m                                      0          1            0      1
      = func ();                               1 (0)      0            0      0
    int n;                                     0          1            0      1
    n = 1;                                     1          0            0      1

#ifdef A_VERY_NICE_VARIABLE

    int j = 0;
    cout << j << endl ;
#endif

    return 0;                                  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

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



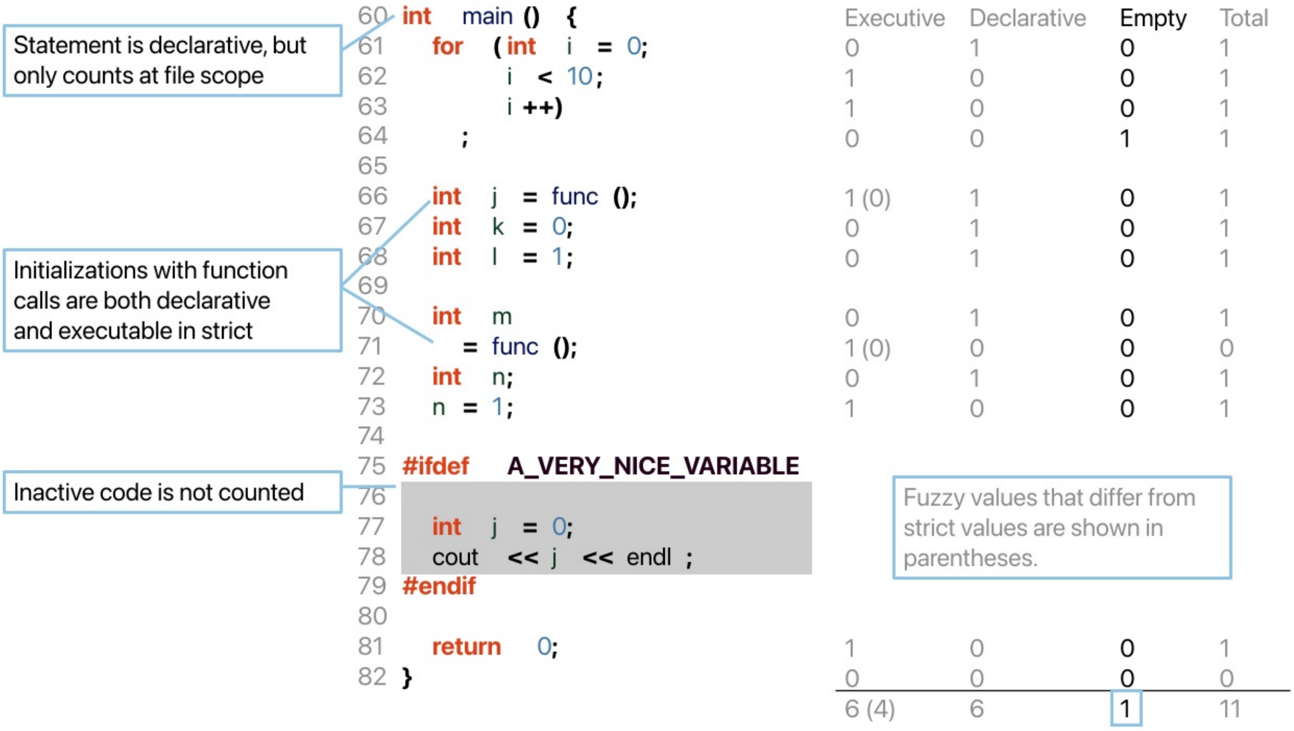
Statement is declarative, but only counts at file scope

Initializations with function calls are both declarative and executable in strict

Inactive code is not counted

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

	Executive	Declarative	Empty	Total
`int main () {`				
`    for ( int i = 0;`	0	1	0	1
`        i < 10;`	1	0	0	1
`        i ++)`	1	0	0	1
`      ;`	0	0	1	1
`    int j = func ();`	1 (0)	1	0	1
`    int k = 0;`	0	1	0	1
`    int l = 1;`	0	1	0	1
`    int m`	0	1	0	1
`      = func ();`	1 (0)	0	0	0
`    int n;`	0	1	0	1
`    n = 1;`	1	0	0	1
`#ifdef A_VERY_NICE_VARIABLE`				
`    int j = 0;`				
`    cout << j << endl ;`				
`#endif`				
`    return 0;`	1	0	0	1
`}`	0	0	0	0
	6 (4)	6	1	11



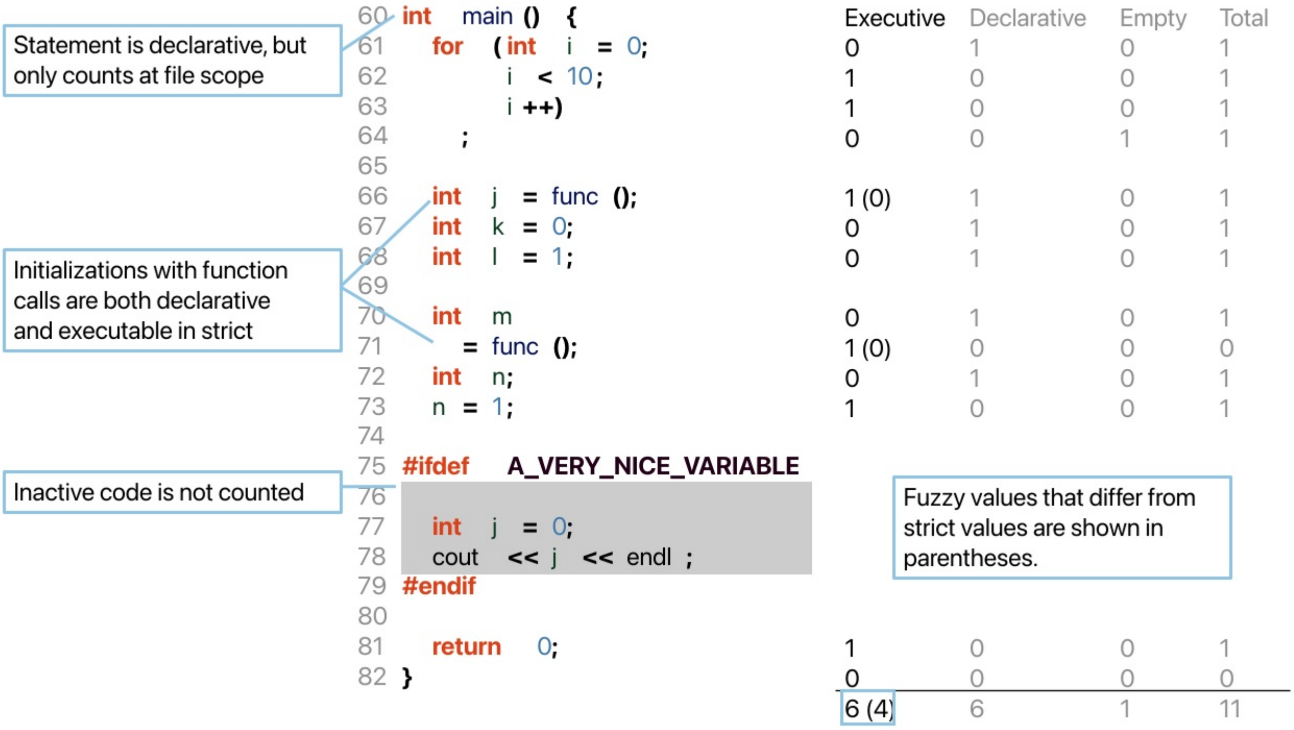
Statement is declarative, but only counts at file scope

Initializations with function calls are both declarative and executable in strict

Inactive code is not counted

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

```
                                           Executive  Declarative  Empty  Total
int main () {
    for ( int i = 0;                       0          1            0      1
          i < 10;                          1          0            0      1
          i ++)                            1          0            0      1
        ;                                  0          0            1      1

    int j = func ();                       1 (0)      1            0      1
    int k = 0;                             0          1            0      1
    int l = 1;                             0          1            0      1

    int m                                  0          1            0      1
        = func ();                         1 (0)      0            0      0
    int n;                                 0          1            0      1
    n = 1;                                 1          0            0      1

#ifdef A_VERY_NICE_VARIABLE

    int j = 0;
    cout << j << endl ;
#endif

    return 0;                              1          0            0      1
}                                          0          0            0      0
                                           ----------------------------------
                                           6 (4)      6            1      11
```



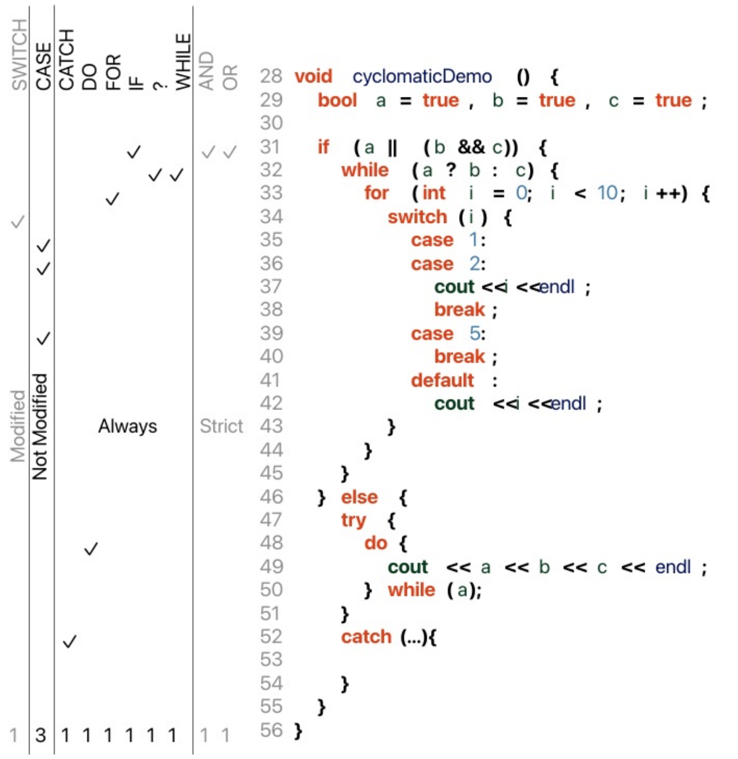
```
int   main ()  {                      Executive   Declarative   Empty   Total
    for   ( int   i   = 0;            0           1             0       1
              i   < 10;               1           0             0       1
              i ++)                   1           0             0       1
        ;                             0           0             1       1

    int   j   = func  ();             1 (0)       1             0       1
    int   k   = 0;                    0           1             0       1
    int   l   = 1;                    0           1             0       1

    int   m                           0           1             0       1
        = func  ();                   1 (0)       0             0       0
    int   n;                          0           1             0       1
    n  = 1;                           1           0             0       1

#ifdef    A_VERY_NICE_VARIABLE

    int   j   = 0;
    cout   << j   << endl  ;
#endif

    return    0;                      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

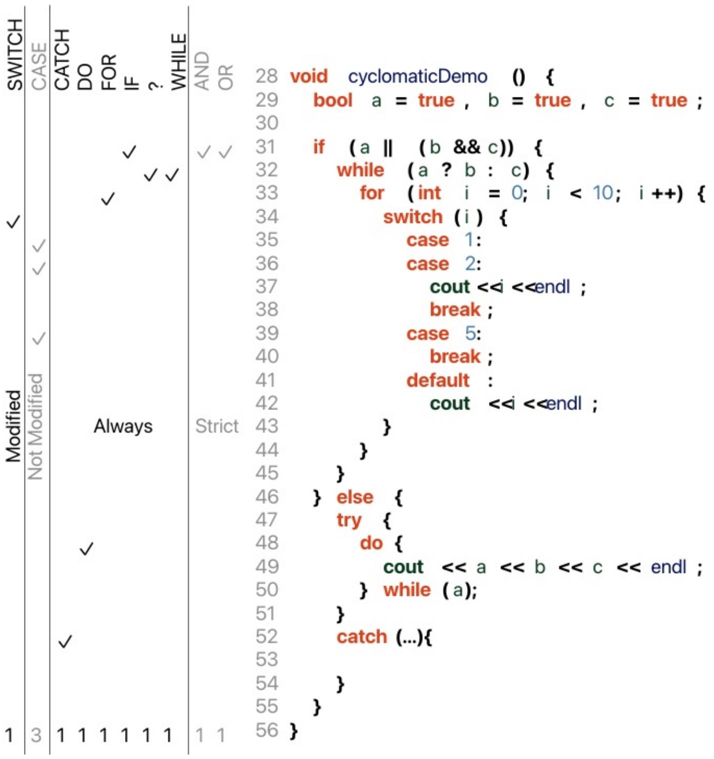
Inactive code is not counted

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



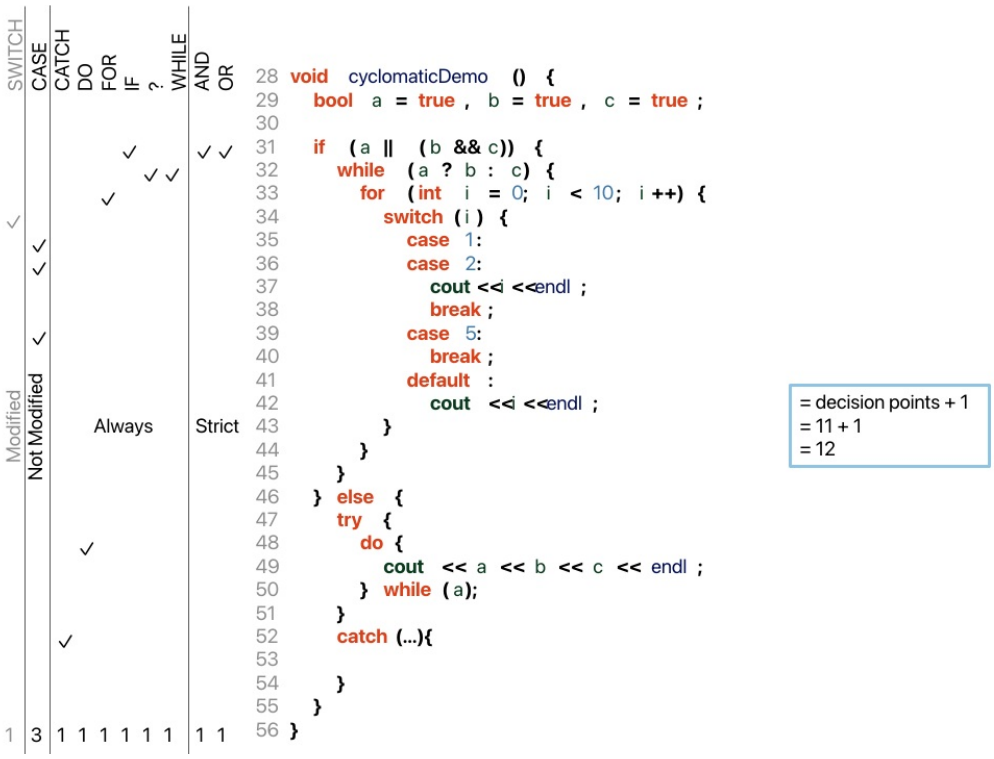
```
void cyclomaticDemo () {
  bool a = true , b = true , c = true ;

  if (a || (b && c)) {
    while (a ? b : c) {
      for (int i = 0; i < 10; i ++) {
        switch (i ) {
          case 1:
          case 2:
            cout <<i <<endl ;
            break ;
          case 5:
            break ;
          default :
            cout <<i <<endl ;
        }
      }
    }
  } else {
    try {
      do {
        cout << a << b << c << endl ;
      } while (a);
    }
    catch (...){

    }
  }
}
```

= decision points + 1
= 9 + 1
= 10



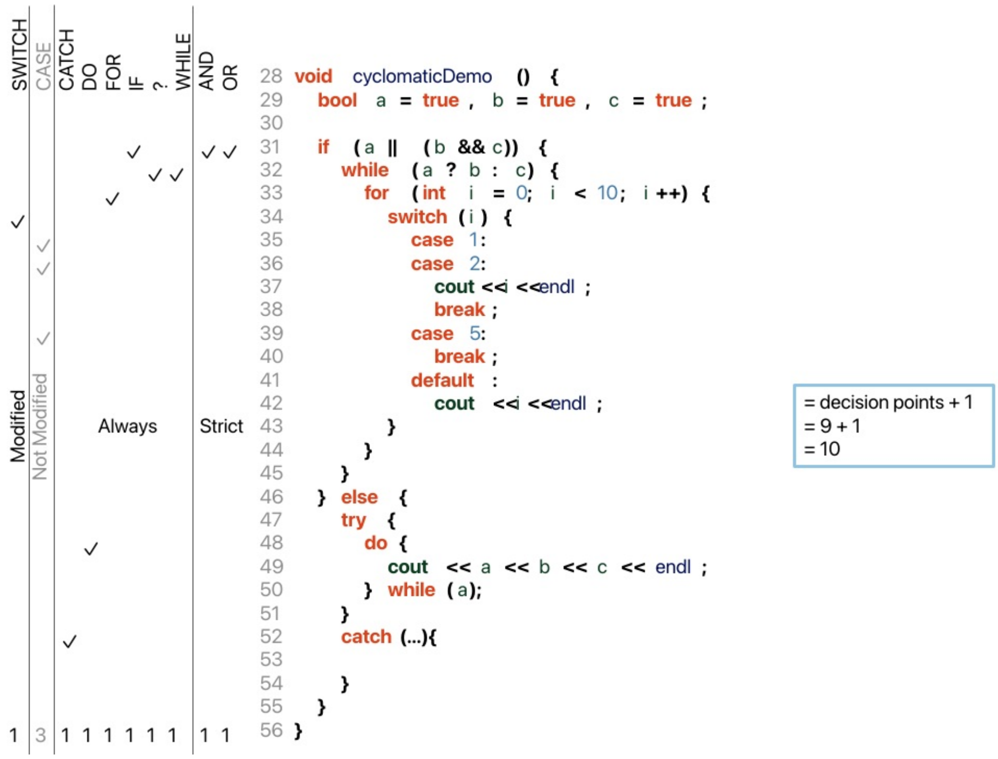
```
void cyclomaticDemo () {
  bool a = true , b = true , c = true ;

  if (a || (b && c)) {
    while (a ? b : c) {
      for (int i = 0; i < 10; i ++) {
        switch (i ) {
          case 1:
          case 2:
            cout <<i <<endl ;
            break ;
          case 5:
            break ;
          default :
            cout <<i <<endl ;
        }
      }
    }
  } else {
    try {
      do {
        cout << a << b << c << endl ;
      } while (a);
    }
    catch (...){

    }
  }
}
```

= decision points + 1
= 7 + 1
= 8



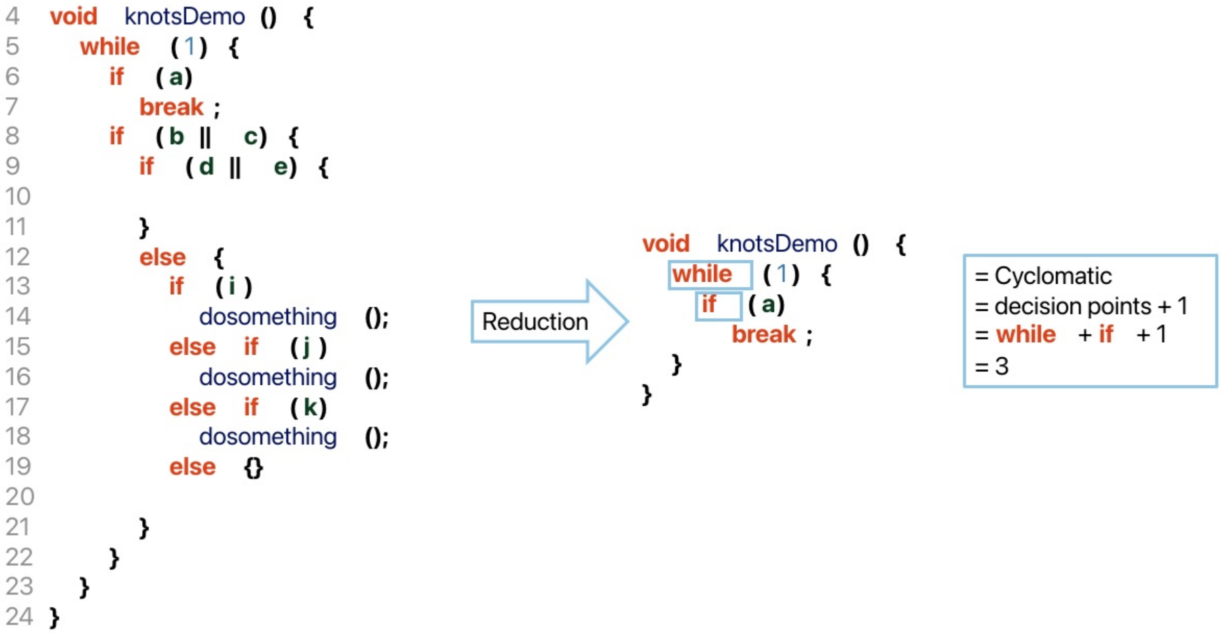
```
void  cyclomaticDemo  ()  {
  bool  a = true ,  b = true ,  c = true ;

  if  ( a ||  ( b  && c))  {
    while  ( a ? b :  c)  {
      for  ( int  i  = 0;  i  < 10;  i ++)  {
        switch  ( i )  {
          case  1:
          case  2:
            cout <<i <<endl ;
            break ;
          case  5:
            break ;
          default  :
            cout  <<i <<endl ;
        }
      }
    }
  }  else  {
    try  {
      do  {
        cout  << a  << b  << c  << endl ;
      }  while  ( a);
    }
    catch (...){

    }
  }
}
```

= decision points + 1
= 11 + 1
= 12



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 ;`
Modified	Not Modified	Always						Strict		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
= 9 + 1
= 10

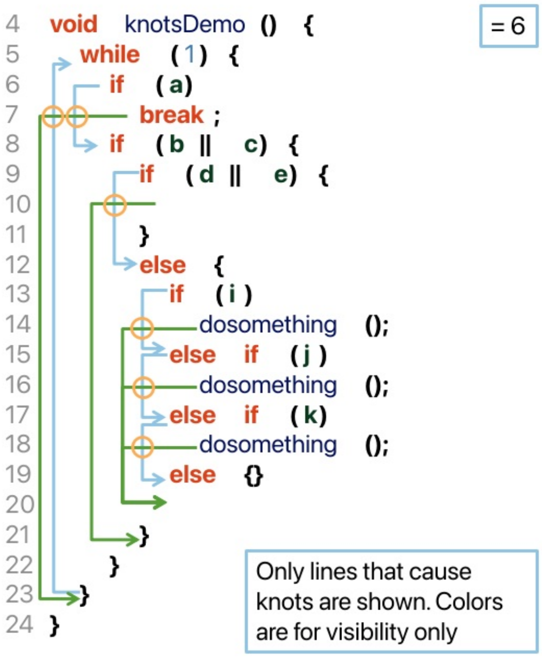


```
void  knotsDemo ()  {
   while  ( 1 )  {
      if  ( a )
         break ;
      if  ( b ||  c)  {
         if  ( d ||  e)  {

         }
         else  {
            if  ( i )
               dosomething  ();
            else  if  ( j )
               dosomething  ();
            else  if  ( k )
               dosomething  ();
            else  {}

         }
      }
   }
}
```

Reduction

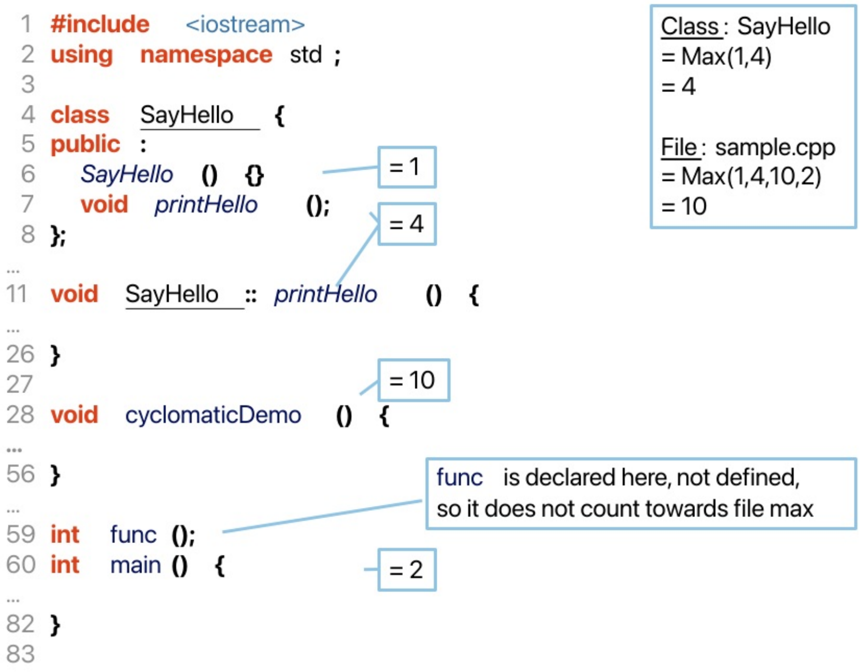
```
void  knotsDemo ()  {
   while  ( 1 )  {
      if  ( a )
         break ;
   }
}
```

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



```
void knotsDemo () {
  while (1) {
    if (a)
      break;
    if (b || c) {
      if (d || e) {

      }
      else {
        if (i)
          dosomething ();
        else if (j)
          dosomething ();
        else if (k)
          dosomething ();
        else {}

      }
    }
  }
}
```

Only lines that cause knots are shown. Colors are for visibility only



```
#include <iostream>
using namespace std ;

class SayHello {
public :
   SayHello () {}
   void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

= 1

= 4

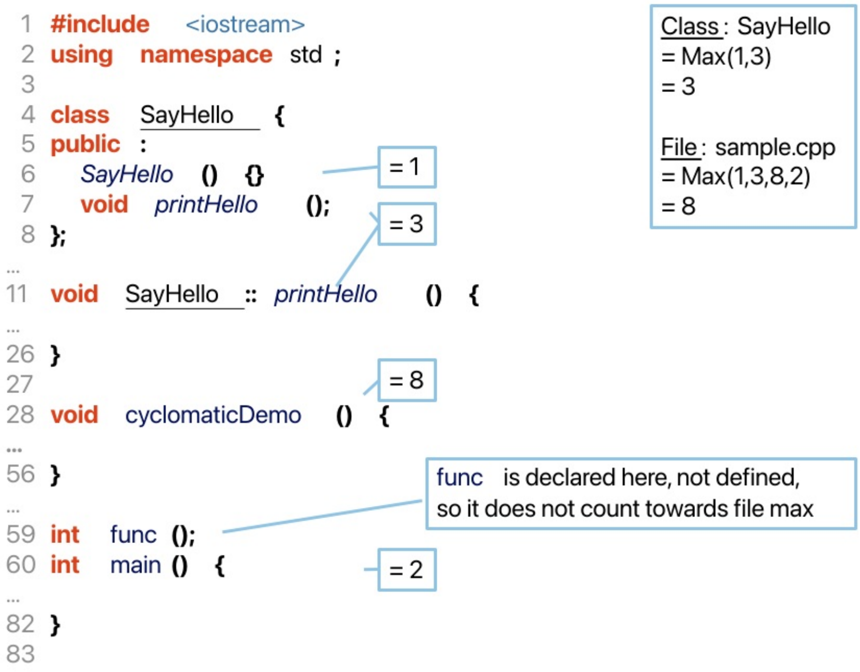
= 10

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

= 2

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

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

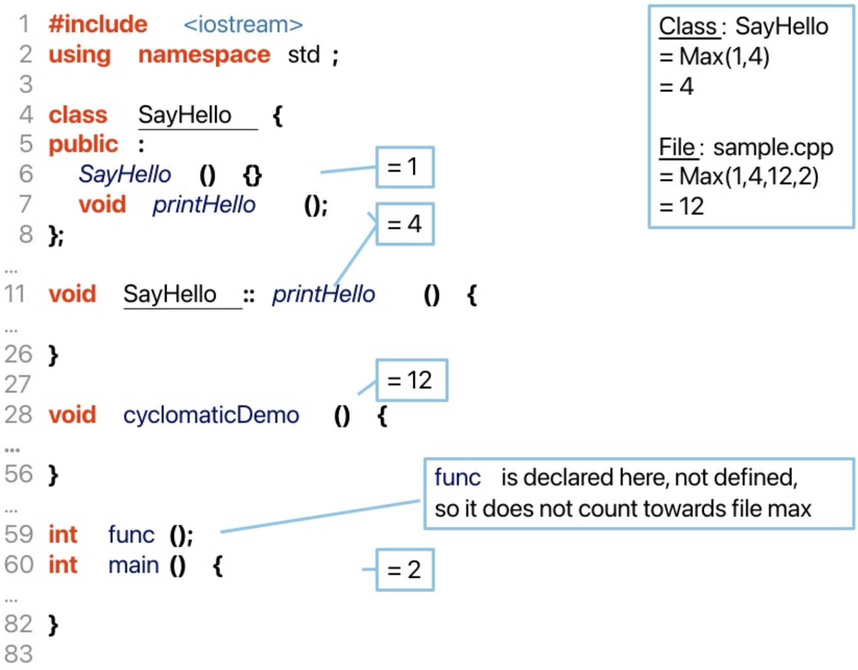


```
#include <iostream>
using namespace std ;

class SayHello {
public :
    SayHello () {}
    void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

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



```
#include <iostream>
using namespace std ;

class SayHello {
public :
   SayHello () {}
   void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

= 1

= 4

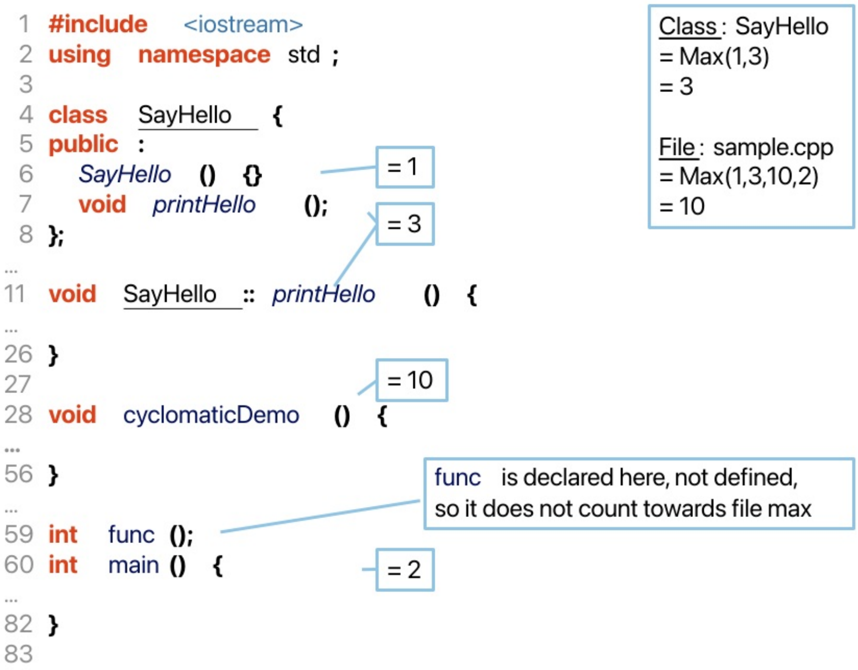
= 12

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

= 2

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

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



```
#include <iostream>
using namespace std ;

class SayHello {
public :
   SayHello () {}
   void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

= 1

= 3

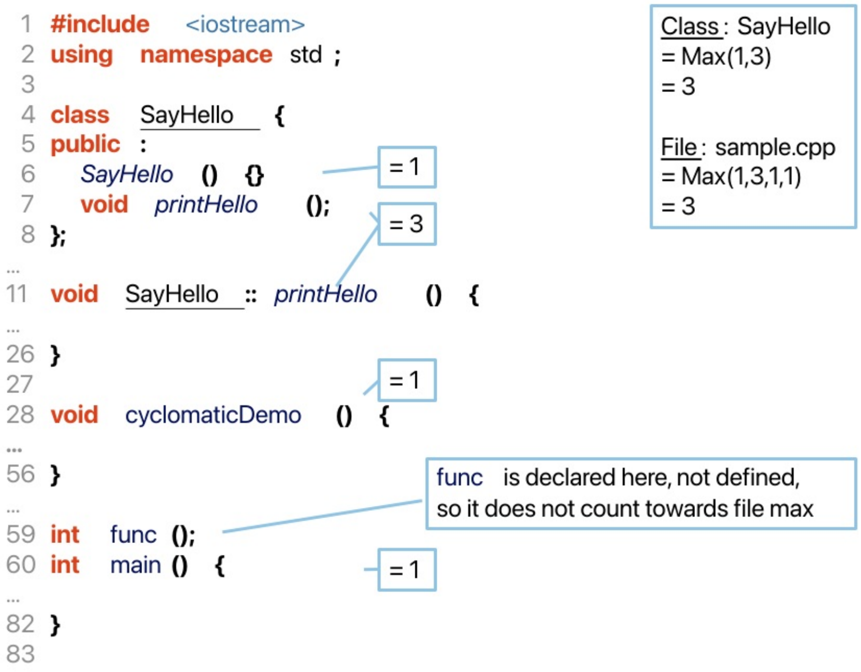
= 10

= 2

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

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

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



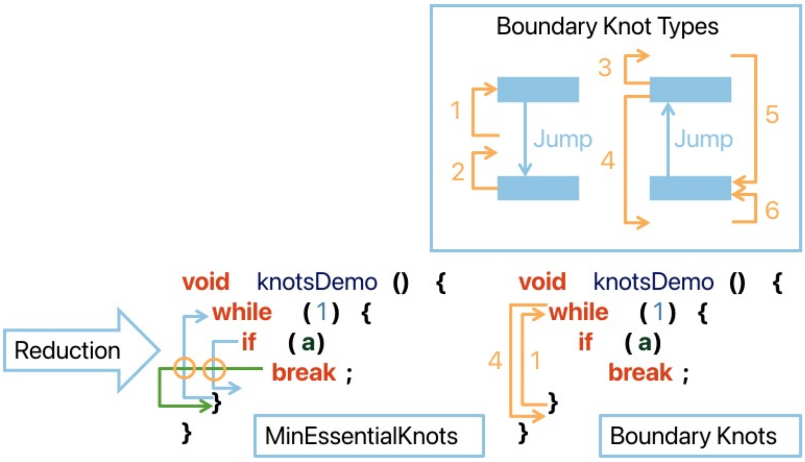
```
#include <iostream>
using namespace std ;

class SayHello {
public :
   SayHello () {}
   void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

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

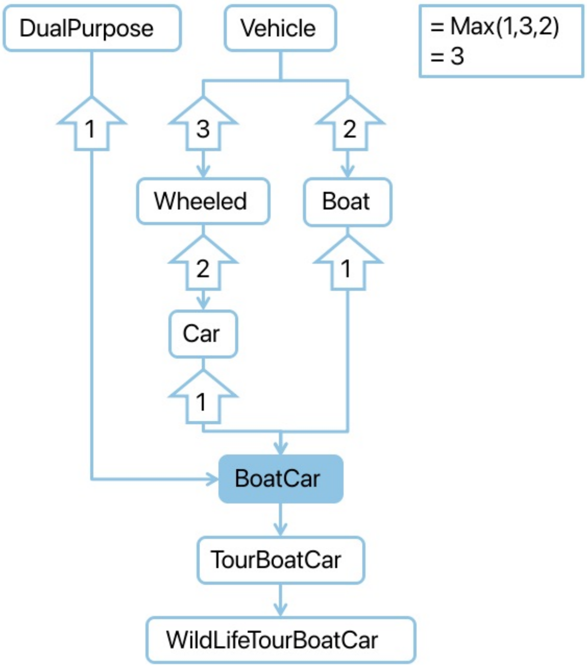
```
void  knotsDemo ()  {
    while  (1)  {
        if  (a)
            break ;
        if  (b ||  c)  {
            if  (d ||  e)  {

            }
            else  {
                if  (i)
                    dosomething  ();
                else  if  (j)
                    dosomething  ();
                else  if  (k)
                    dosomething  ();
                else  {}

            }
        }
    }
}
```

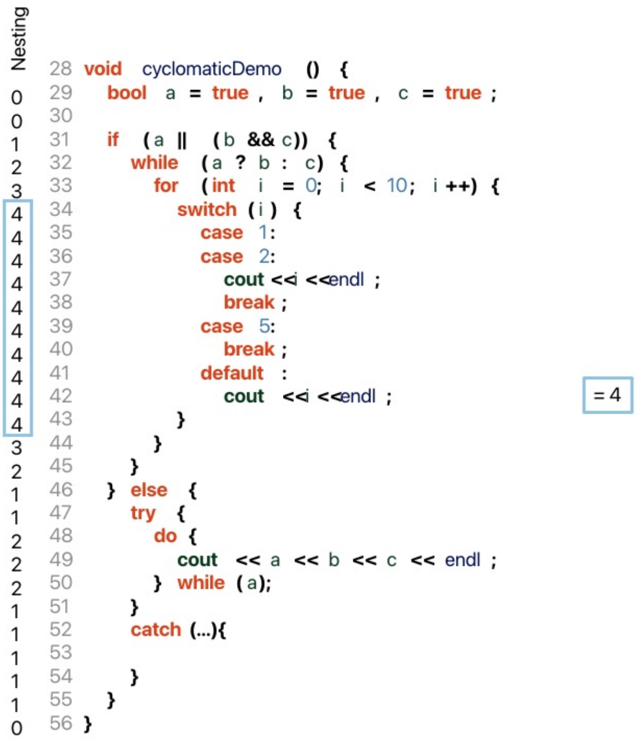


```
void  knotsDemo ()  {
    while  (1)  {
        if  (a)
            break ;
    }
}
```

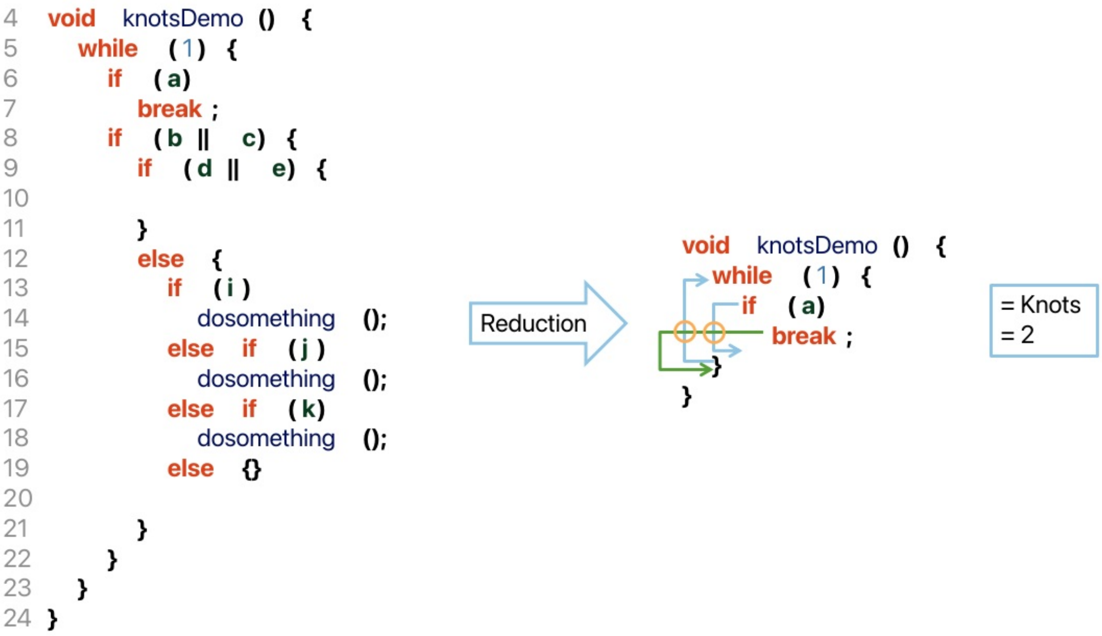
```
void  knotsDemo ()  {
    while  (1)  {
        if  (a)
            break ;
    }
}
```

= MinEssentialKnots + (Boundary Knots/ 2)
= 2 + (2/2)
= 3


DualPurpose
Vehicle
= Max(1,3,2)
= 3
1
3
2
Wheeled
Boat
2
1
Car
1
BoatCar
TourBoatCar
WildLifeTourBoatCar

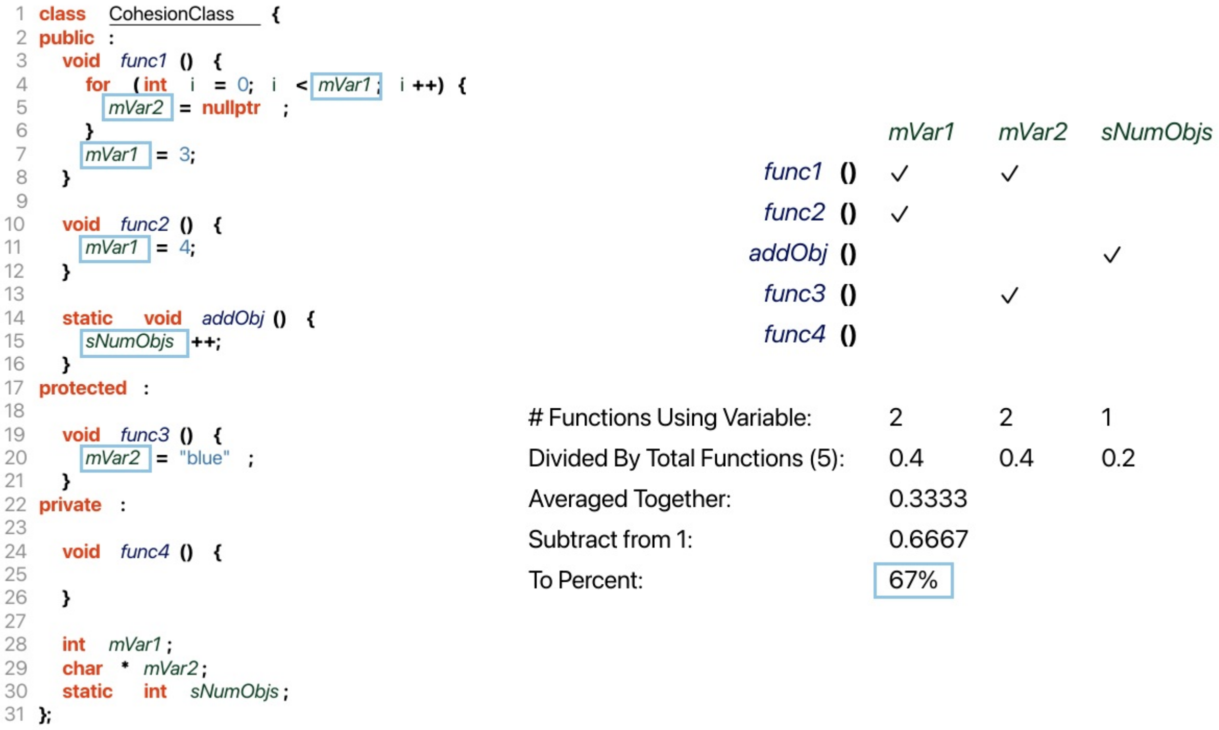


```
Nesting
  void  cyclomaticDemo  ()  {
0   bool  a = true ,  b = true ,  c = true ;
0
1   if  (a ||  (b && c))  {
2     while  (a ? b :  c) {
3       for  (int  i  = 0;  i  < 10;  i ++) {
4         switch (i )  {
4           case  1:
4           case  2:
4             cout <<i <<endl ;
4             break ;
4           case  5:
4             break ;
4           default  :
4             cout  <<i <<endl ;                        = 4
4         }
3       }
2     }
1   } else  {
1     try  {
2       do {
2         cout  << a  << b << c  << endl ;
2       } while (a);
1     }
1     catch (...){
1
1     }
1   }
0 }
```



```
void  knotsDemo () {
   while  (1) {
      if  (a)
         break ;
      if  (b ||  c) {
         if  (d ||  e) {

         }
         else  {
            if  (i)
               dosomething  ();
            else if  (j)
               dosomething  ();
            else if  (k)
               dosomething  ();
            else {}

         }
      }
   }
}
```

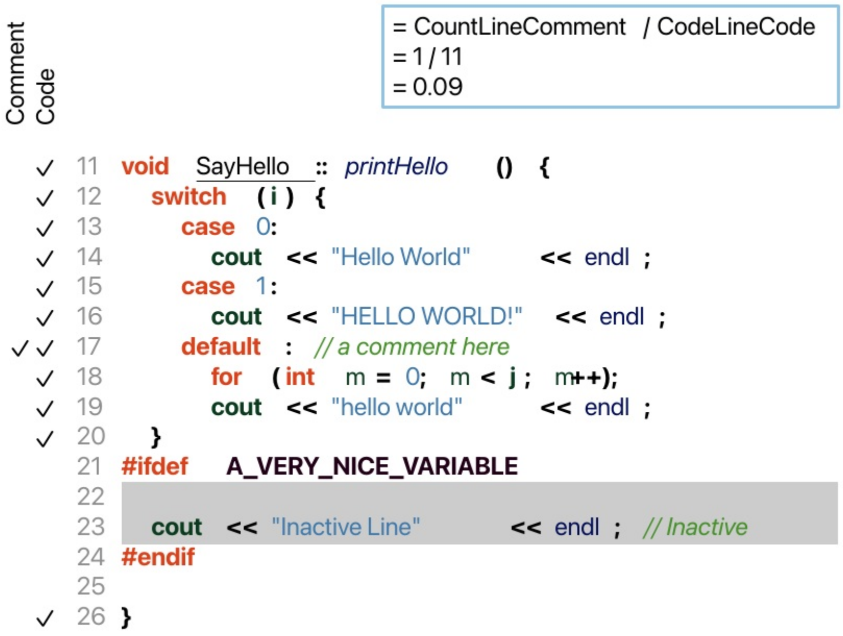
```
void  knotsDemo () {
   while  (1) {
      if  (a)
         break ;
   }
}
```



```
class CohesionClass {
public :
  void func1 () {
    for (int i = 0; i < mVar1; i ++) {
      mVar2 = nullptr ;
    }
    mVar1 = 3;
  }

  void func2 () {
    mVar1 = 4;
  }

  static void addObj () {
    sNumObjs ++;
  }
protected :

  void func3 () {
    mVar2 = "blue" ;
  }
private :

  void func4 () {

  }

  int mVar1 ;
  char * mVar2 ;
  static int sNumObjs ;
};
```

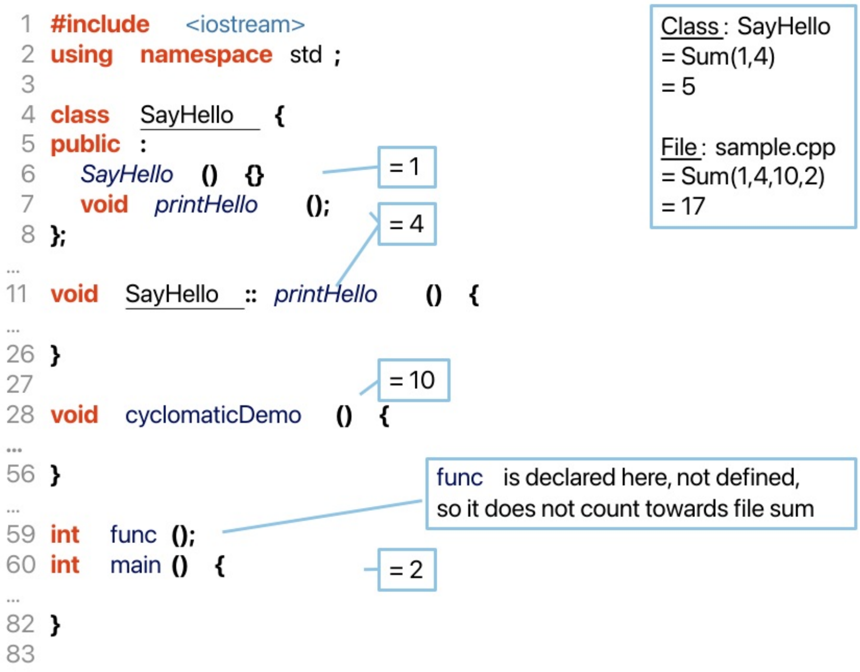
	mVar1	mVar2	sNumObjs
func1 ()	✓	✓	
func2 ()	✓		
addObj ()			✓
func3 ()		✓	
func4 ()			

# 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%		



= CountLineComment / CodeLineCode
= 1 / 11
= 0.09

```
Comment Code
   ✓     void SayHello :: printHello () {
   ✓       switch (i) {
   ✓         case 0:
   ✓           cout << "Hello World" << endl ;
   ✓         case 1:
   ✓           cout << "HELLO WORLD!" << endl ;
✓  ✓         default : // a comment here
   ✓           for ( int m = 0; m < j ; m++);
   ✓           cout << "hello world" << endl ;
   ✓       }
         #ifdef A_VERY_NICE_VARIABLE

           cout << "Inactive Line" << endl ; // Inactive
         #endif

   ✓     }
```

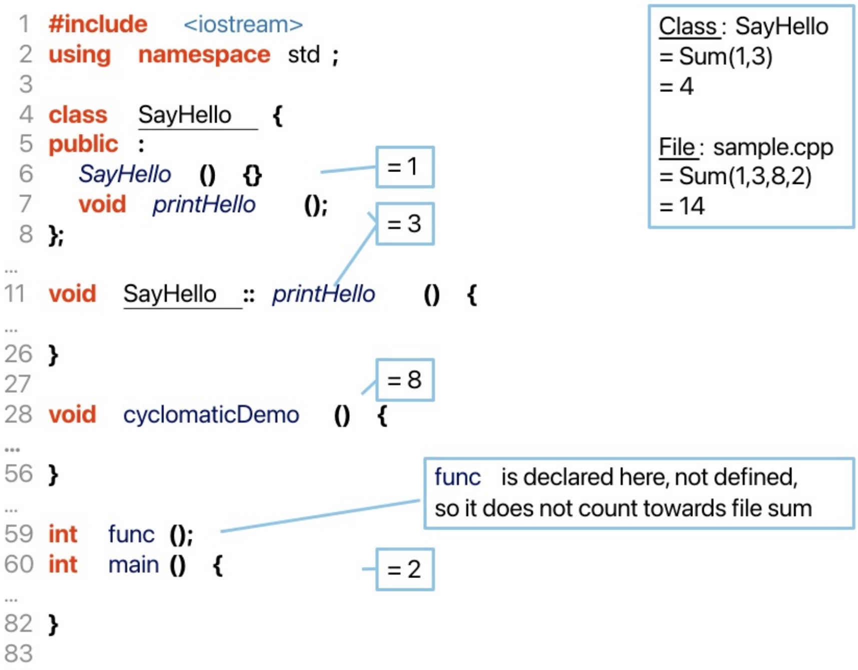


```
#include <iostream>
using namespace std ;

class SayHello {
public :
   SayHello () {}
   void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

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

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

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

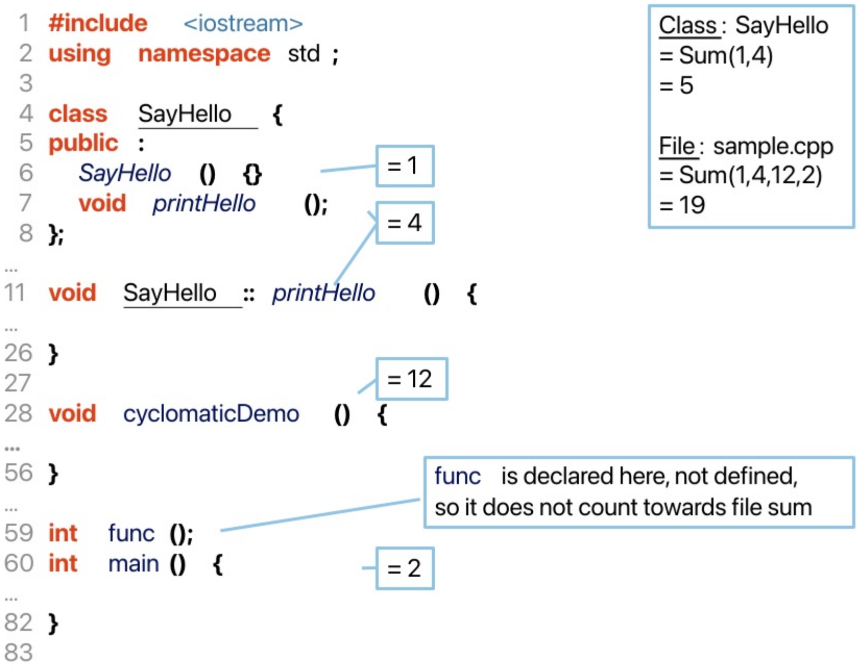


```
#include <iostream>
using namespace std;

class SayHello {
public:
   SayHello() {}
   void printHello();
};
...
void SayHello::printHello() {
...
}

void cyclomaticDemo() {
...
}
...
int func();
int main() {
...
}

```

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

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

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



```
#include   <iostream>
using  namespace  std ;

class  SayHello   {
public  :
   SayHello   ()   {}
   void  printHello       ();
};
...
void  SayHello  ::  printHello       ()   {
...
}

void  cyclomaticDemo    ()   {
...
}
...
int   func  ();
int   main  ()   {
...
}

```

= 1

= 4

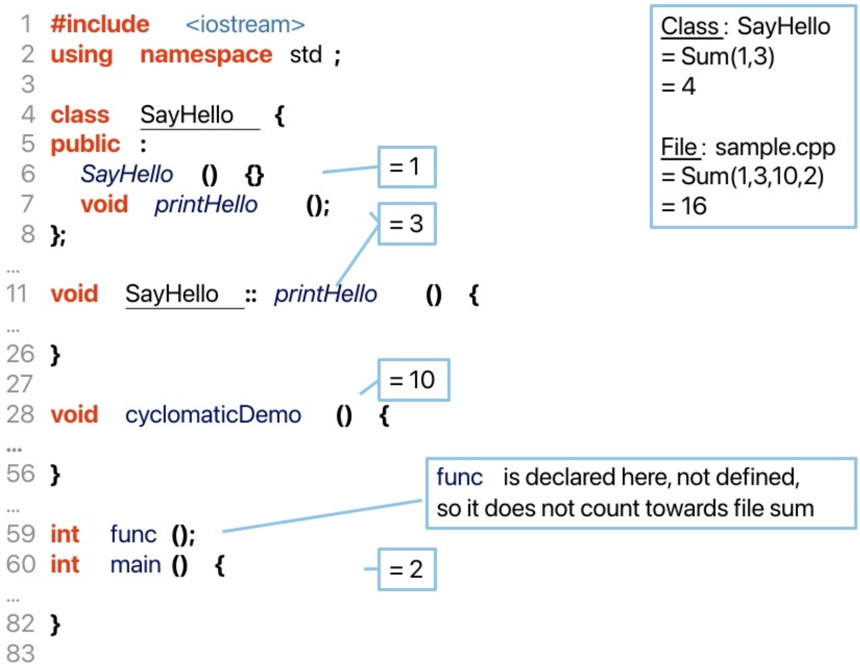
= 12

= 2

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

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

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



```
#include <iostream>
using namespace std ;

class SayHello {
public :
   SayHello () {}
   void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

= 1

= 3

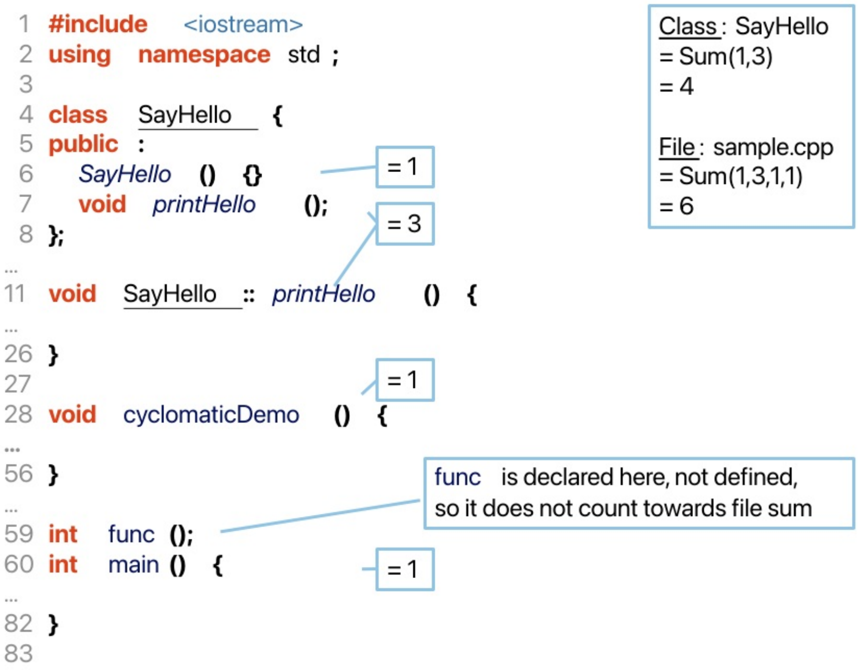
= 10

= 2

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

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

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



```
#include <iostream>
using namespace std ;

class SayHello {
public :
   SayHello () {}
   void printHello ();
};
...
void SayHello :: printHello () {
...
}

void cyclomaticDemo () {
...
}
...
int func ();
int main () {
...
}

```

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