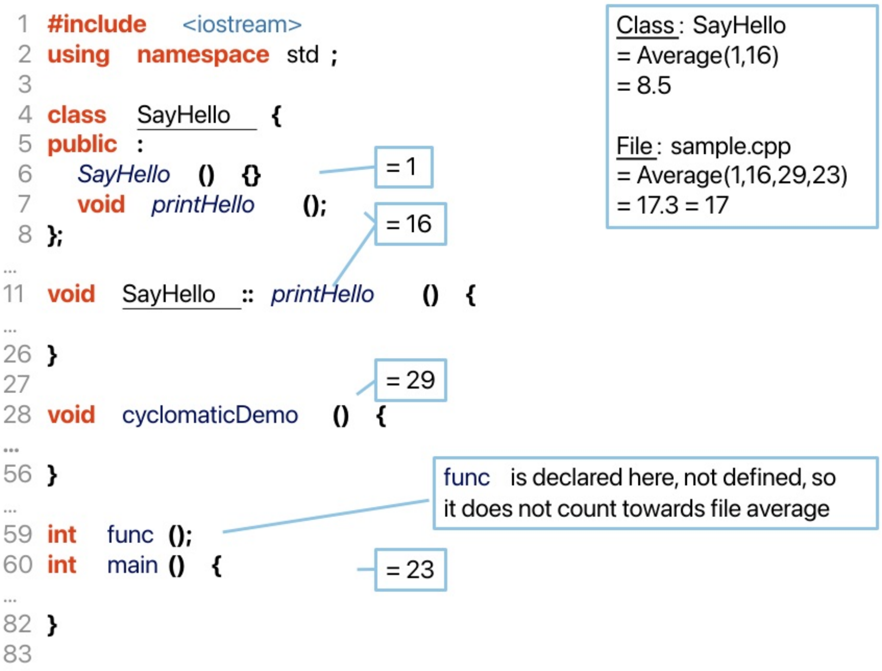
Understand Software Metrics

Table of Contents

Average Lines

API Name: AvgCountLine
Description: Average number of lines for all nested functions or methods.
Available For:

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



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

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

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

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

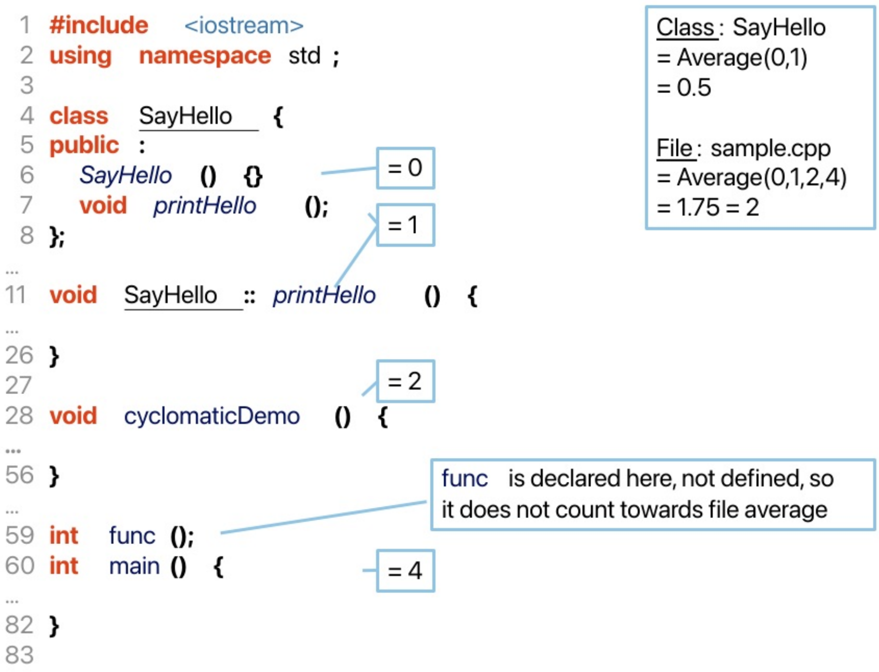
File : sample.cpp
= Average(1,16,29,23)
= 17.3 = 17

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

Average Blank Lines

API Name: AvgCountLineBlank
Description: Average number of blank lines for all nested functions or methods.
Available For:

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



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

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

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

```

= 0

= 1

= 2

= 4

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

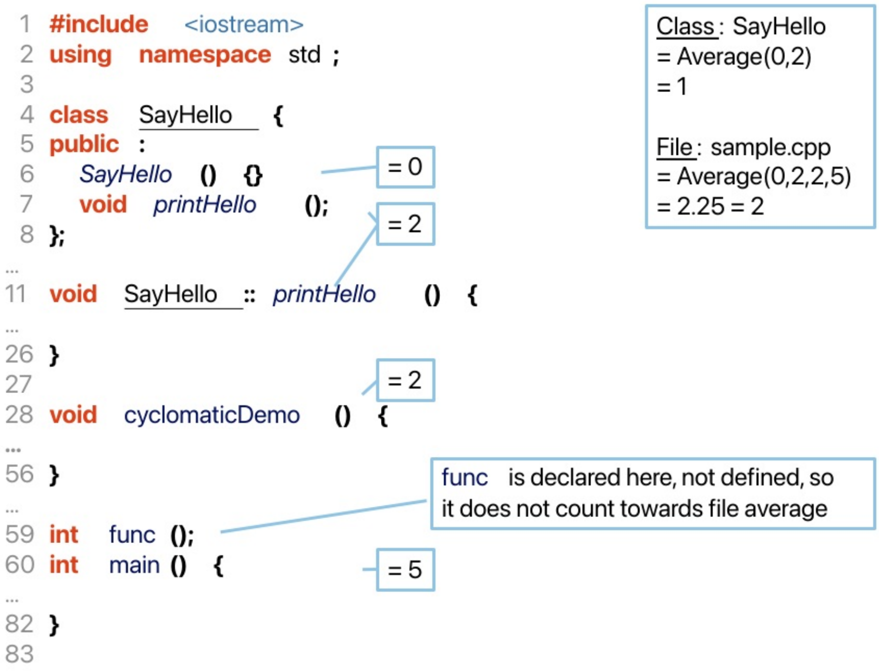
Class: SayHello
= Average(0,1)
= 0.5

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

Average Blank Lines (Includes Inactive)

API Name: AvgCountLineBlankWithInactive
Description: Average number of blank lines for all nested functions or methods, including inactive regions.
Available For:

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



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

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

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

```

= 0

= 2

= 2

= 5

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

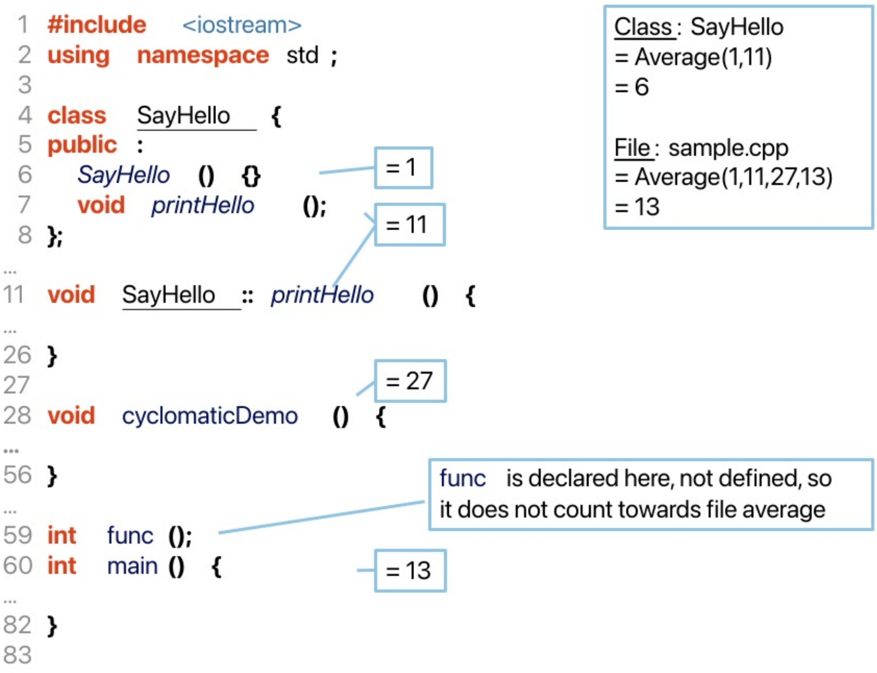
Class: SayHello
= Average(0,2)
= 1

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

Average Code Lines

API Name: AvgCountLineCode
Description: Average number of lines containing source code for all nested functions or methods.
Available For:

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



```
#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

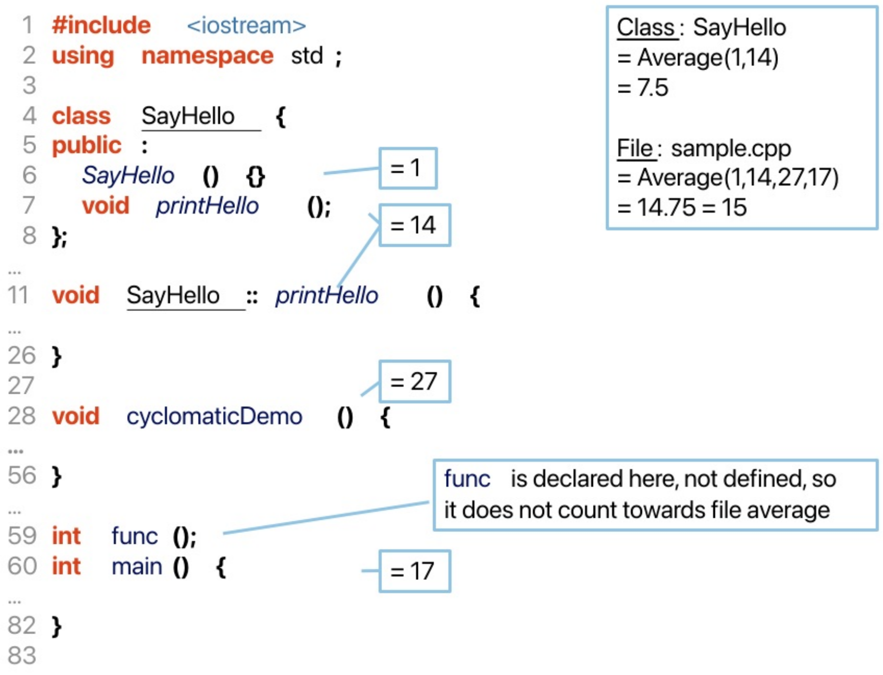
File: sample.cpp
= Average(1,11,27,13)
= 13

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

Average Code Lines (Includes Inactive)

API Name: AvgCountLineCodeWithInactive
Description: Average number of lines containing source code for all nested functions or methods, including inactive regions.
Available For:

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



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

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

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

```

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

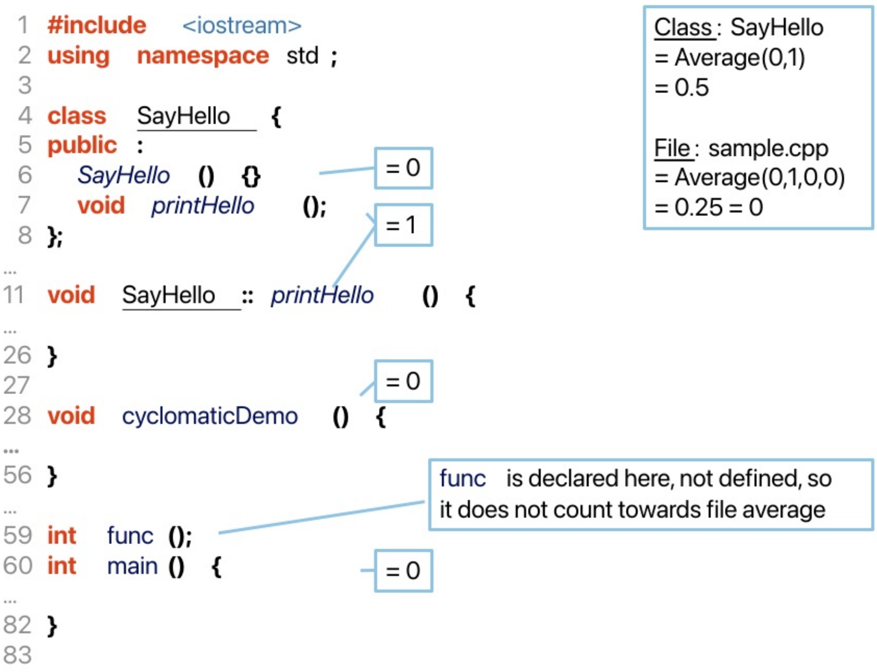
File: sample.cpp
= Average(1,14,27,17)
= 14.75 = 15

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

Average Comment Lines

API Name: AvgCountLineComment
Description: Average number of lines containing comment for all nested functions or methods.
Available For:

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



```
#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

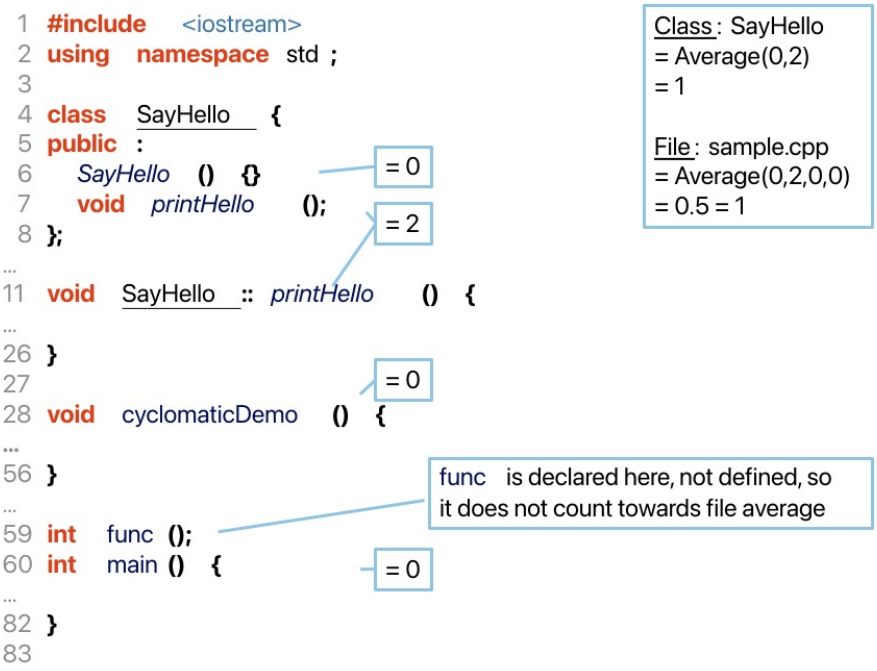
Class: SayHello
= Average(0,1)
= 0.5

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

Average Comment Lines (Includes Inactive)

API Name: AvgCountLineCommentWithInactive
Description: Average number of lines containing comment for all nested functions or methods, including inactive regions.
Available For:

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



```
#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

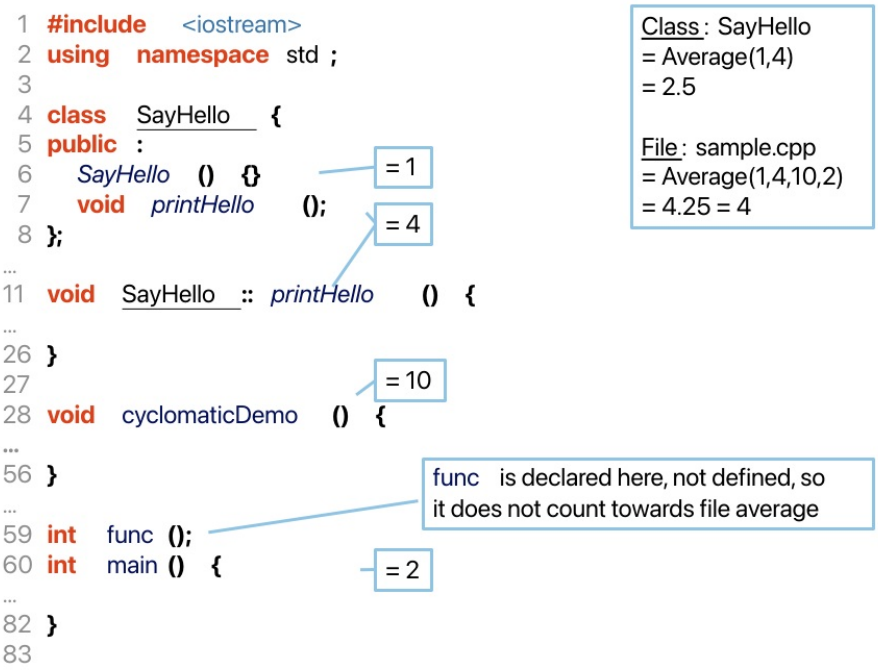
Class: SayHello
= Average(0,2)
= 1

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

Average Cyclomatic Complexity

API Name: AvgCyclomatic
Description: Average cyclomatic complexity for all nested functions or methods.
Available For:

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



```
#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

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

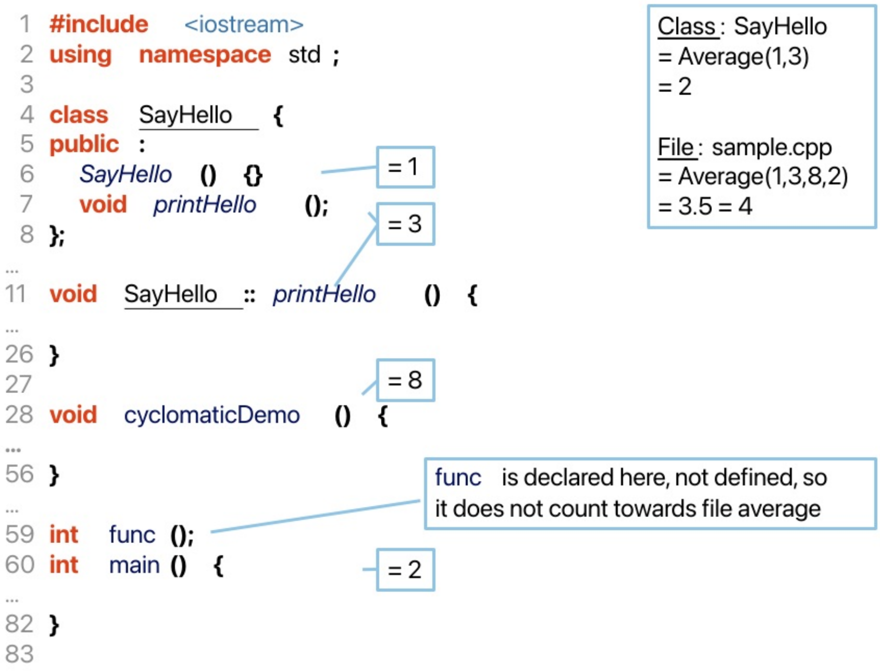
func is declared here, not defined, so it does not count towards file average

Average Modified Cyclomatic Complexity

API Name: AvgCyclomaticModified
Description: Average modified cyclomatic complexity for all nested functions or methods.
Available For:

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

- **C#:** Project,File,Class,Struct
- **Fortran:** Project,File
- **Java:** Project,File,Class,Interface
- **Jovial:** Project,File
- **Pascal:** Project,File,Class,Interface,Compunit
- **Python:** Project,File,Class
- **VHDL:** Project,File,Architecture
- **Web:** Project,File,PHP Class,PHP Interface



```
#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

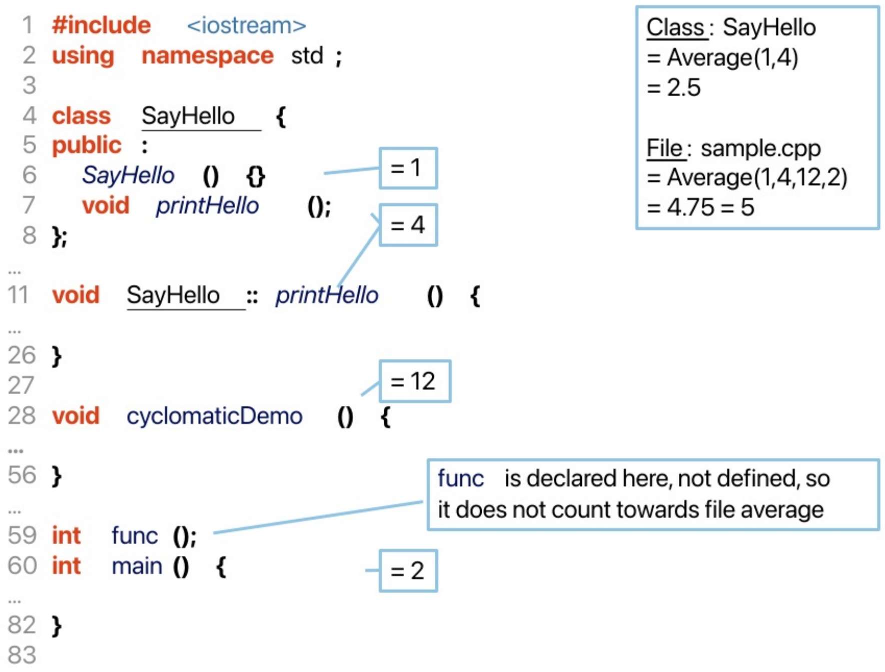
File: sample.cpp
= Average(1,3,8,2)
= 3.5 = 4

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

Average Strict Cyclomatic Complexity

API Name: AvgCyclomaticStrict
Description: Average strict cyclomatic complexity for all nested functions or methods.
Available For:

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



```
#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

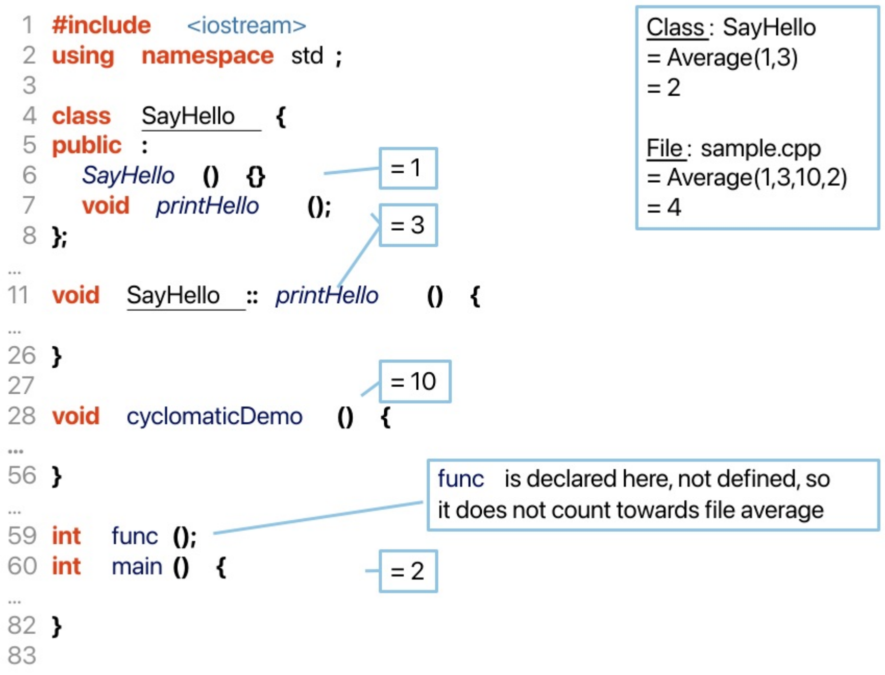
File : sample.cpp
= Average(1,4,12,2)
= 4.75 = 5

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

Average Strict Modified Cyclomatic Complexity

API Name: AvgCyclomaticStrictModified
Description: Average strict modified cyclomatic complexity for all nested functions or methods.
Available For:

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



```
#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

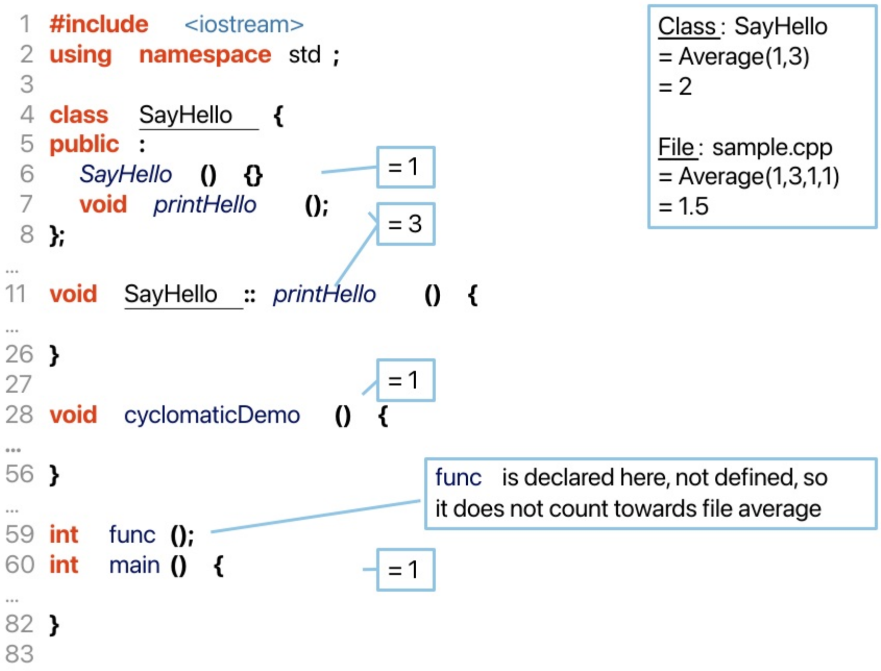
File: sample.cpp
= Average(1,3,10,2)
= 4

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

Average Essential Complexity

API Name: AvgEssential
Description: Average Essential complexity for all nested functions or methods.
Available For:

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



```
#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(1,3,1,1)
= 1.5

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

Average Strict Modified Essential Complexity

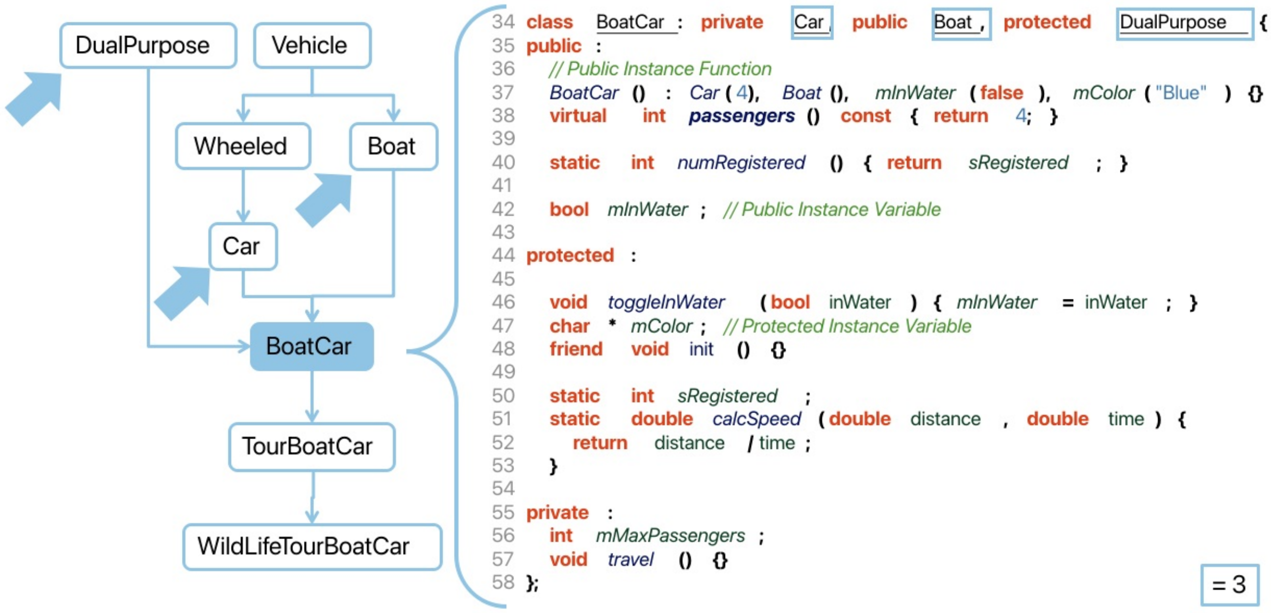
API Name: AvgEssentialStrictModified
Description: Average strict modified essential complexity for all nested functions or methods.
Available For:

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

Base Classes

API Name: CountClassBase
Research Name:IFANIN
Description: Number of immediate base classes. [aka IFANIN]
Available For:

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



```
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 () {}
};
```

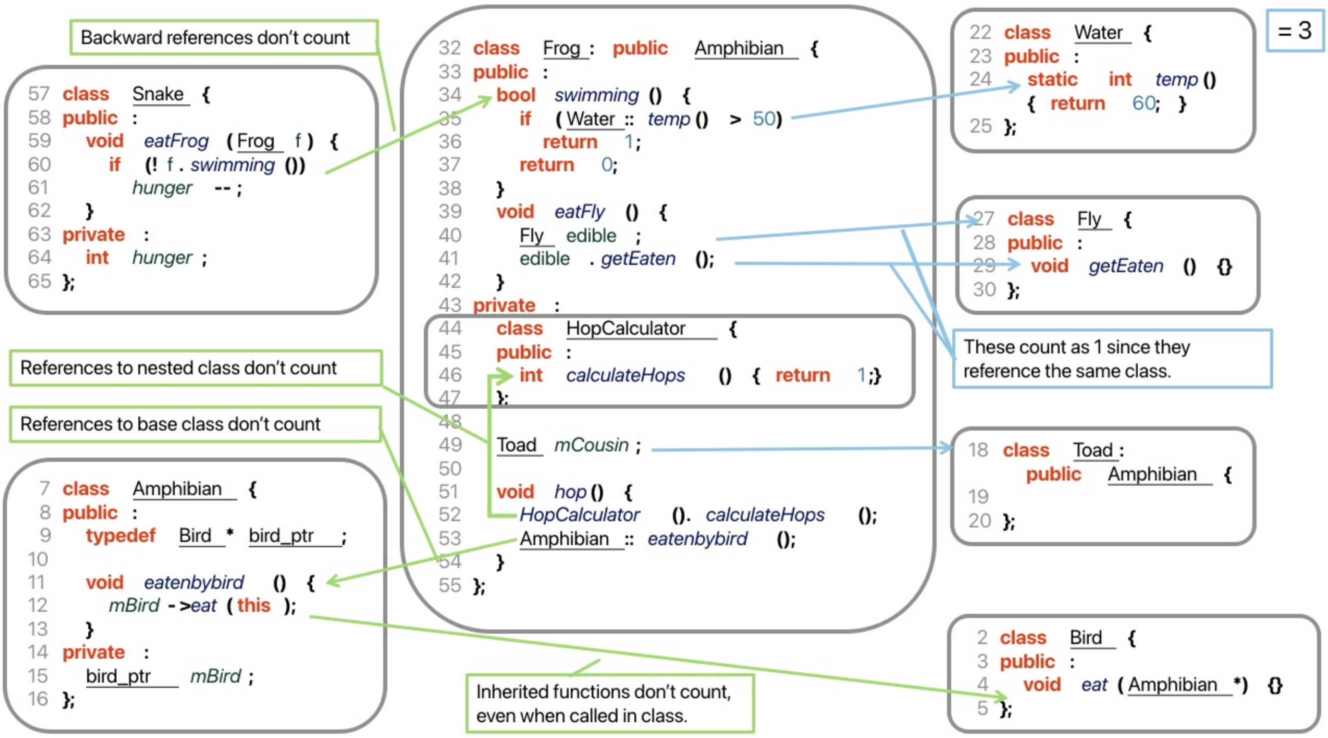
Coupled Classes

API Name: CountClassCoupled
Research Name: Chidamber & Kemerer - Coupling Between Objects (CBO)
Description: Number of other classes coupled to. [aka CBO (coupling between object classes)]

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

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

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



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 *) {}
};
```

Coupled Classes Modified

API Name: CountClassCoupledModified
Description: Number of other non-standard classes coupled to.
Available For:

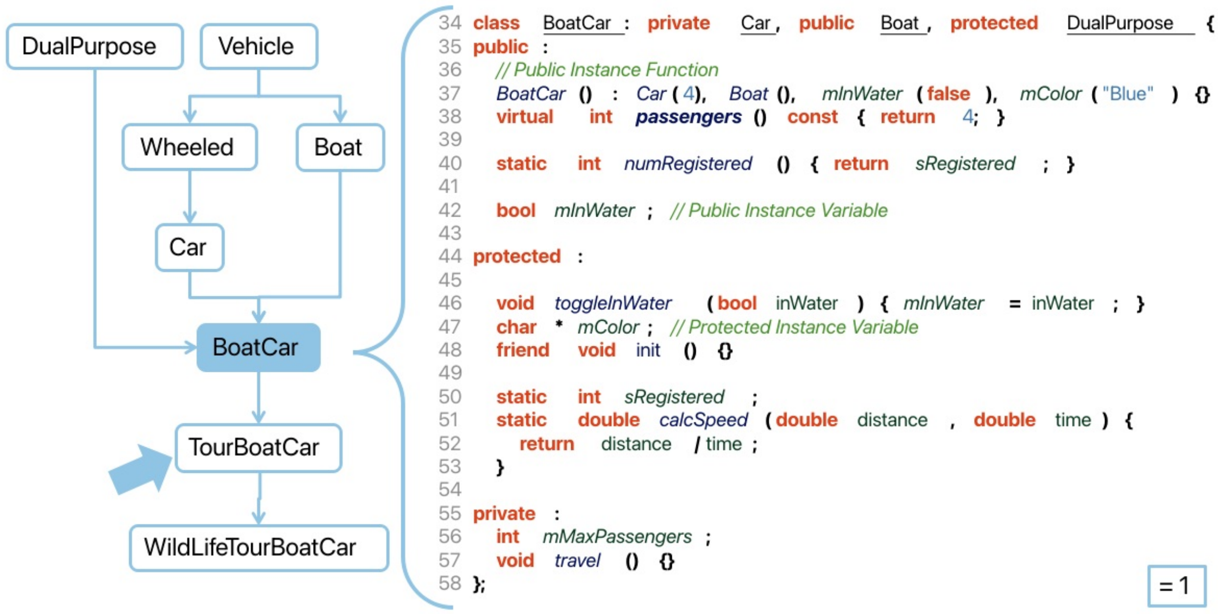
- **Basic:** Class,Struct
- **C#:** Class,Struct
- **Java:** Class,Interface
- **Pascal:** Class,Interface
- **Python:** Class

Derived Classes

API Name: CountClassDerived
Research Name: Chidamber & Kemerer - Number of Children (NOC)
Description: Number of immediate subclasses. [aka NOC (number of children)]

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

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



```
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 () {}
};
```

Classes

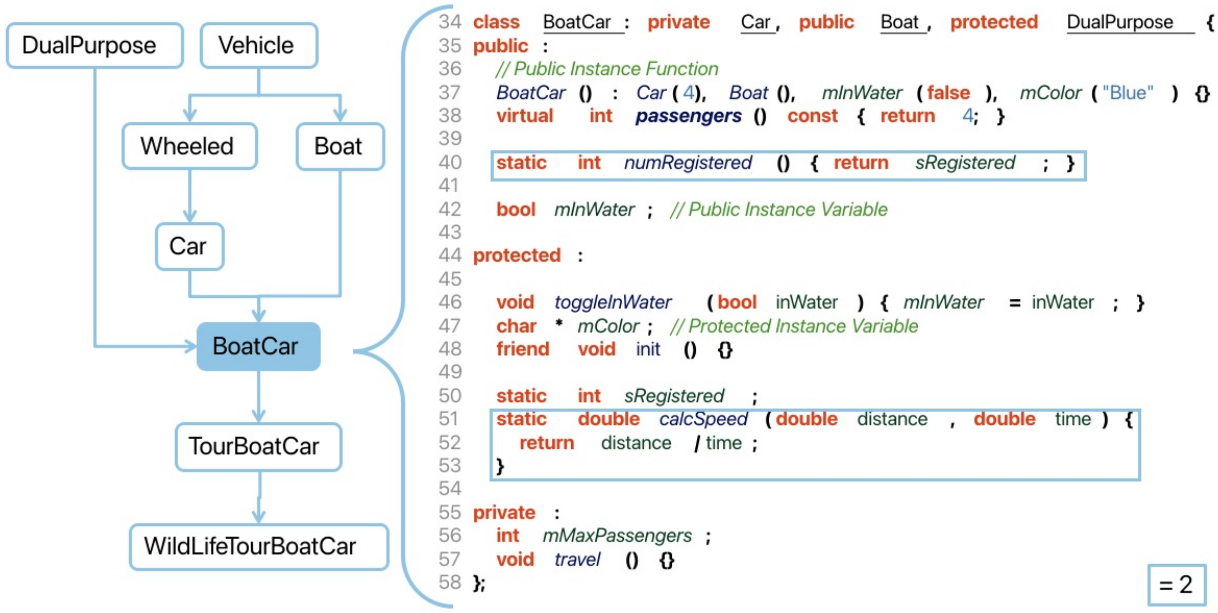
API Name: CountDeclClass
Description: Number of classes.
Available For:

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

Class Methods

API Name: CountDeclClassMethod
Description: Number of class methods.
Available For:

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

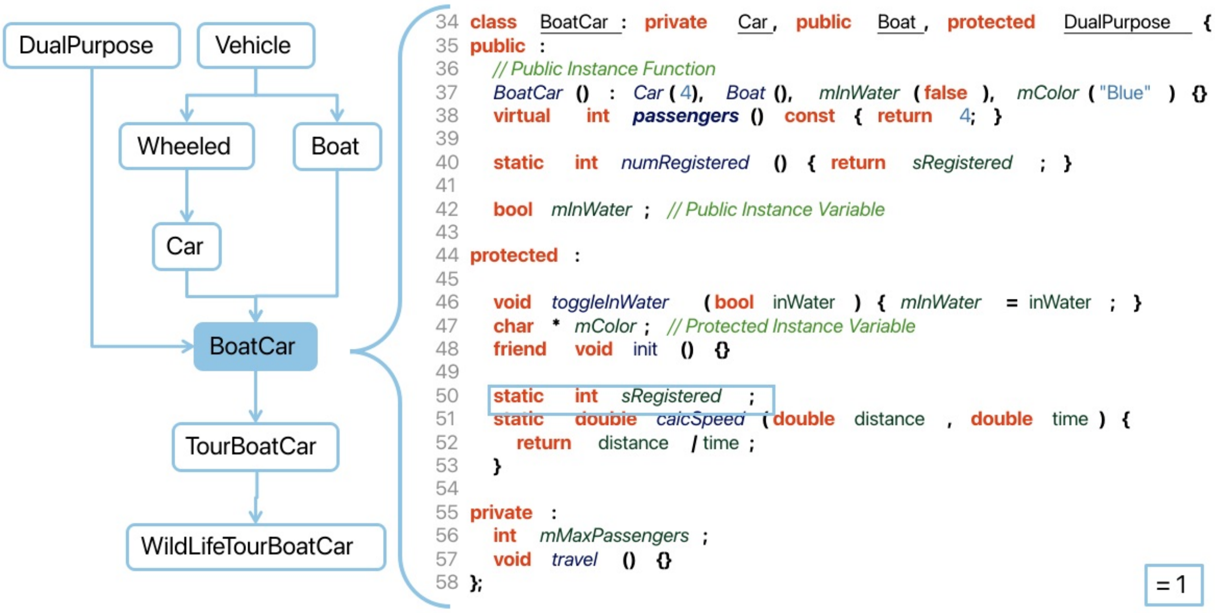


```
class BoatCar : private Car, public Boat, protected DualPurpose {
public :
  // Public Instance Function
  BoatCar () : Car (4), Boat (), mInWater (false), mColor ("Blue") {}
  virtual int passengers () const { return 4; }

  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 Variables

API Name: CountDeclClassVariable
Research Name: Lorenz & Kidd - Number of Variables (NV)
Description: Number of class variables. [aka NV]
Available For:

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



```
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 () {}
};
```

Executable Units

API Name: CountDeclExecutableUnit
Description: Number of program units with executable code.
Available For:

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

Files

API Name: CountDeclFile
Description: Number of files.
Available For:

- **Ada:** Project
- **Basic:** Project
- **C/C++:** Project
- **C#:** Project
- **Fortran:** Project
- **Java:** Project
- **Jovial:** Project

- **Pascal:** Project
- **Python:** Project
- **VHDL:** Project
- **Web:** Project

Code Files

API Name: CountDeclFileCode
Description: Number of code files.
Available For:

- **C/C++:** Project

Header Files

API Name: CountDeclFileHeader
Description: Number of header files.
Available For:

- **C/C++:** Project

Functions

API Name: CountDeclFunction
Description: Number of functions.
Available For:

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

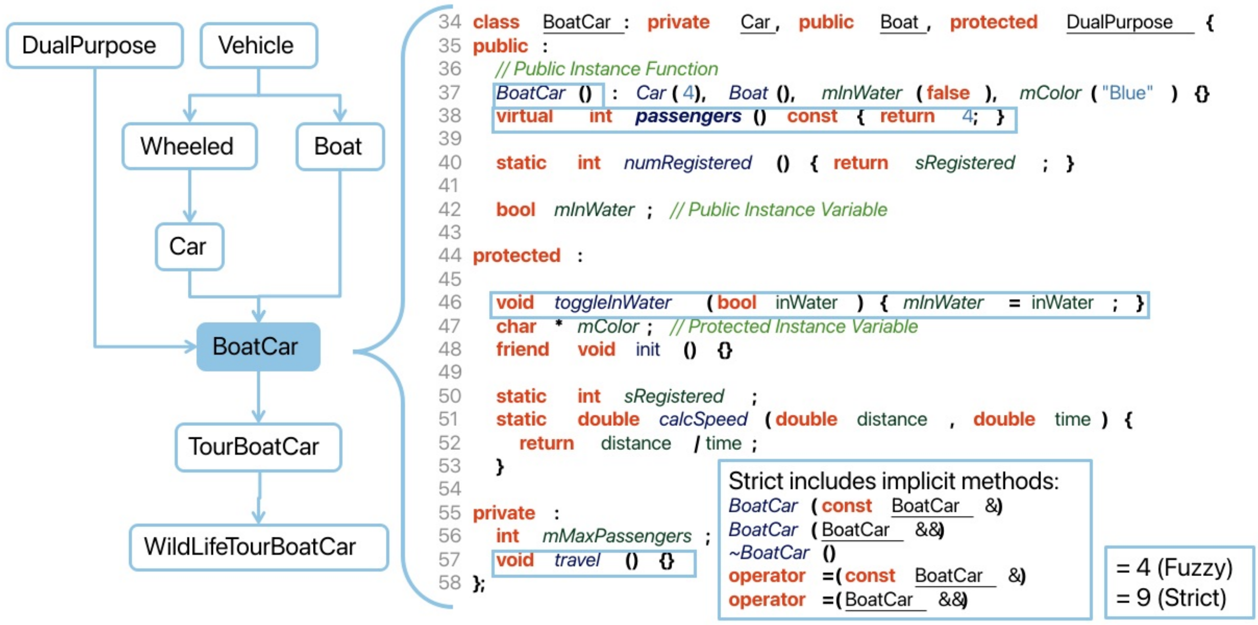
Instance Methods

API Name: CountDeclInstanceMethod
Research Name: Number of Instance Methods (NIM)
Description: Number of instance methods. [aka NIM]

Methods defined in a class that are only accessable through an object of that class.
Available For:

- **Basic:** Project,Class,Struct
- **C/C++:** Project,Class,Struct,Union
- **C#:** Project,Class,Struct
- **Java:** Project,File,Class,Interface

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



```
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 &&)
```

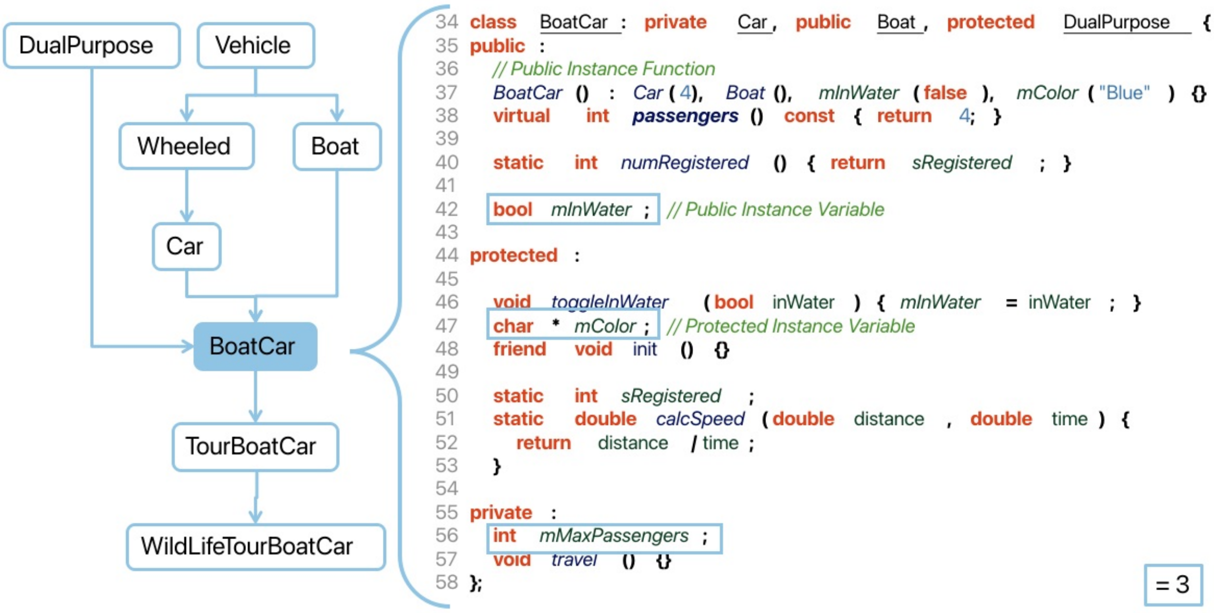
= 4 (Fuzzy)
= 9 (Strict)

Instance Variables

API Name: CountDeclInstanceVariable
Research Name: Number of Instance Variables (NIV)
Description: Number of instance variables. [aka NIV]

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

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



```
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 () {}
};
```

Internal Instance Variables

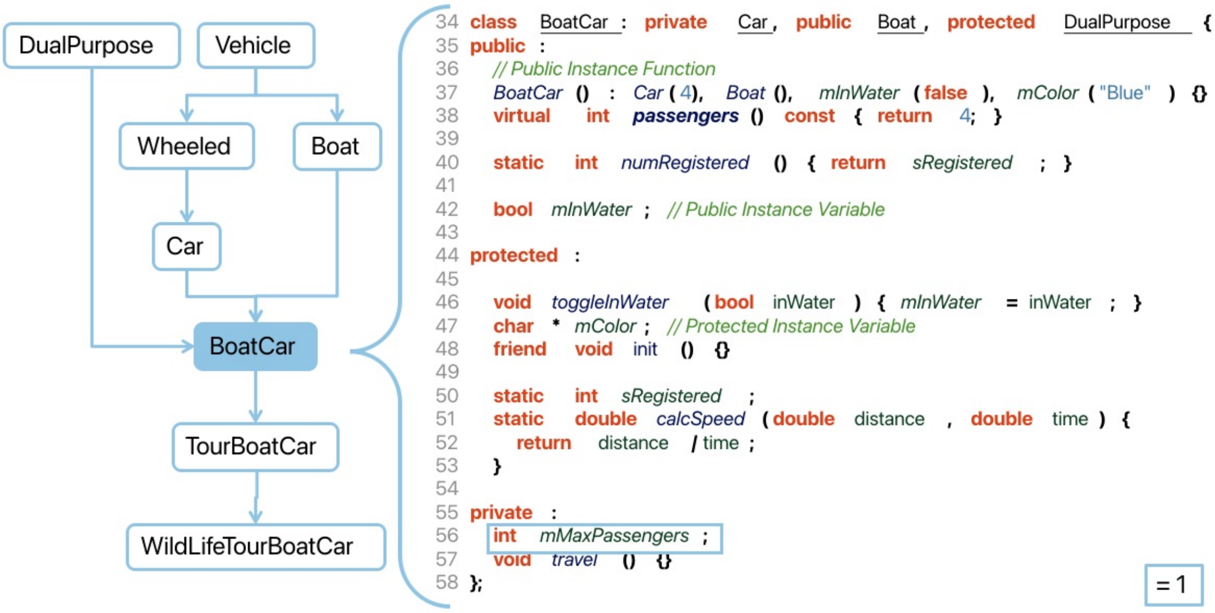
API Name: CountDeclInstanceVariableInternal
Description: Number of internal instance variables.
Available For:

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

Private Instance Variables

API Name: CountDeclInstanceVariablePrivate
Description: Number of private instance variables.
Available For:

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



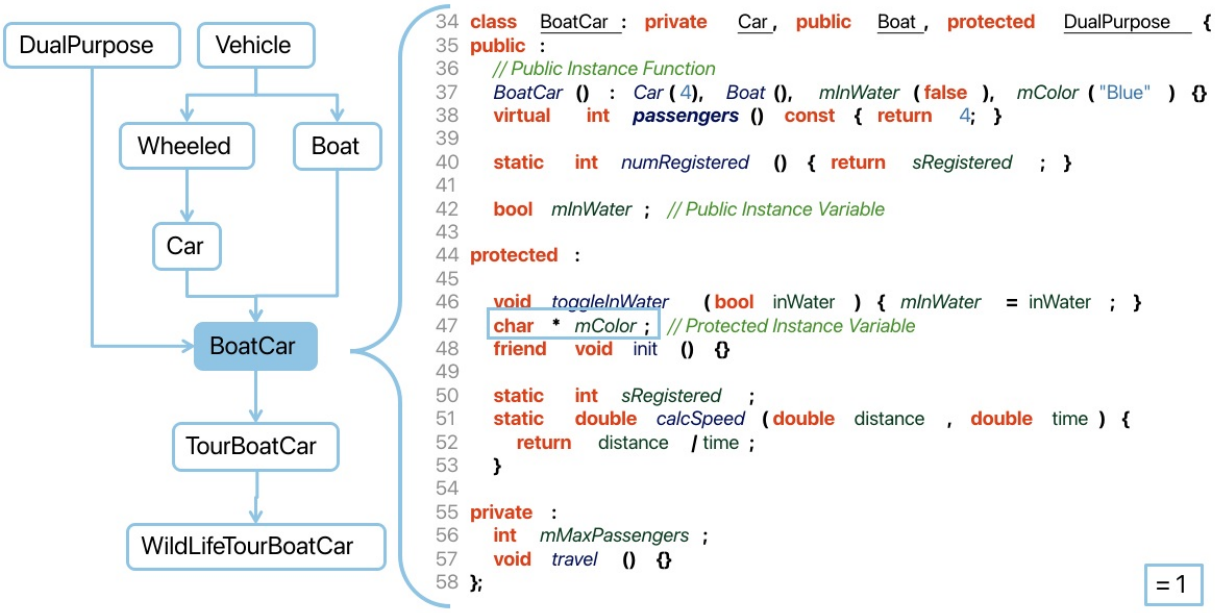
```
class BoatCar : private Car, public Boat, protected DualPurpose {
public :
  // Public Instance Function
  BoatCar () : Car ( 4), Boat (), mInWater ( false ), mColor ( "Blue" ) {}
  virtual int passengers () const { return 4; }

  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

Protected Instance Variables

API Name: CountDeclInstanceVariableProtected
Description: Number of protected instance variables.
Available For:

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



```
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

Protected Internal Instance Variables

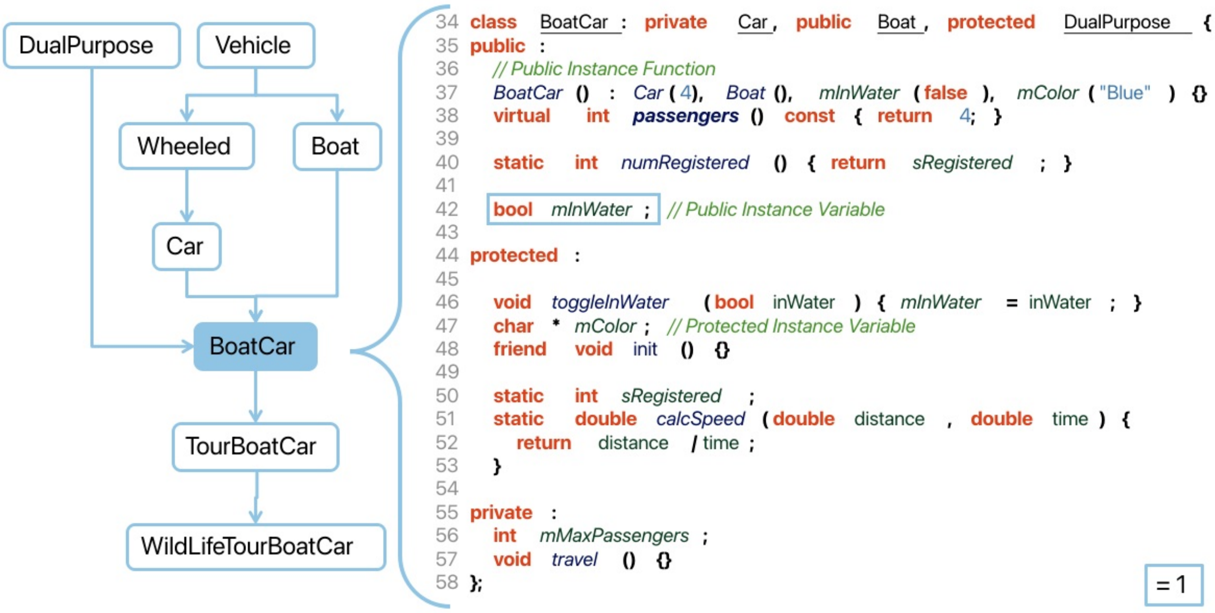
API Name: CountDeclInstanceVariableProtectedInternal
Description: Number of protected internal instance variables.
Available For:

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

Public Instance Variables

API Name: CountDeclInstanceVariablePublic
Description: Number of public instance variables.
Available For:

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



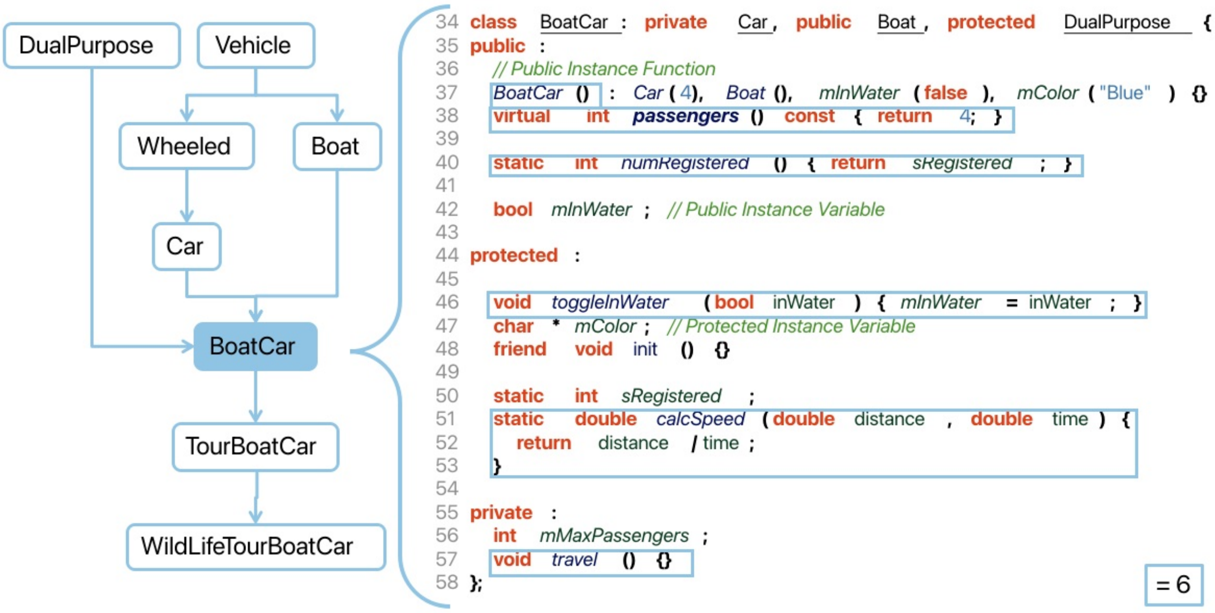
```
class BoatCar : private Car, public Boat, protected DualPurpose {
public :
  // Public Instance Function
  BoatCar () : Car (4), Boat (), mInWater (false ), mColor ("Blue" ) {}
  virtual int passengers () const { return 4; }

  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 () {}
};
```

Methods

API Name: CountDeclMethod
Research Name: Chidamber & Kemerer - Weighted Methods per Class (WMC)
Description: Number of local (not inherited) methods. [aka WMC (weighted methods per class)]
Available For:

- **Basic:** Project,Module,Class,Struct
- **C/C++:** Project,Class,Struct,Union

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



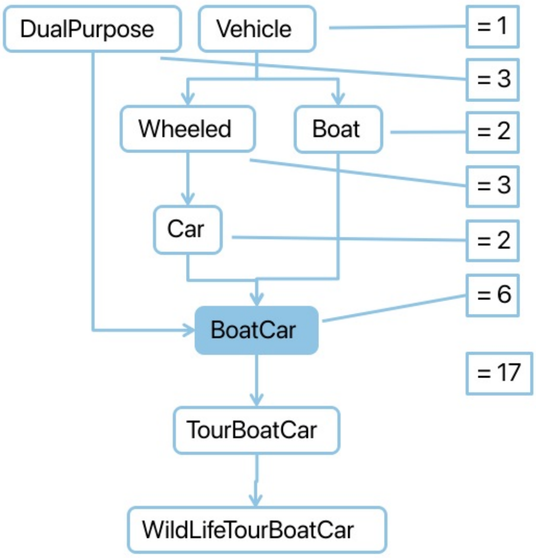
```
class BoatCar : private Car, public Boat, protected DualPurpose {
public :
  // Public Instance Function
  BoatCar () : Car (4), Boat (), mInWater (false), mColor ("Blue") {}
  virtual int passengers () const { return 4; }

  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 () {}
};
```

= 6

All Methods

API Name: CountDeclMethodAll
Research Name: Chidamber & Kemerer - Response for a Class (RFC), Lorenz & Kidd - Number of Methods (NM)
Description: Number of methods, including inherited ones. [aka RFC (response for class), NM (number of methods)]
Available For:

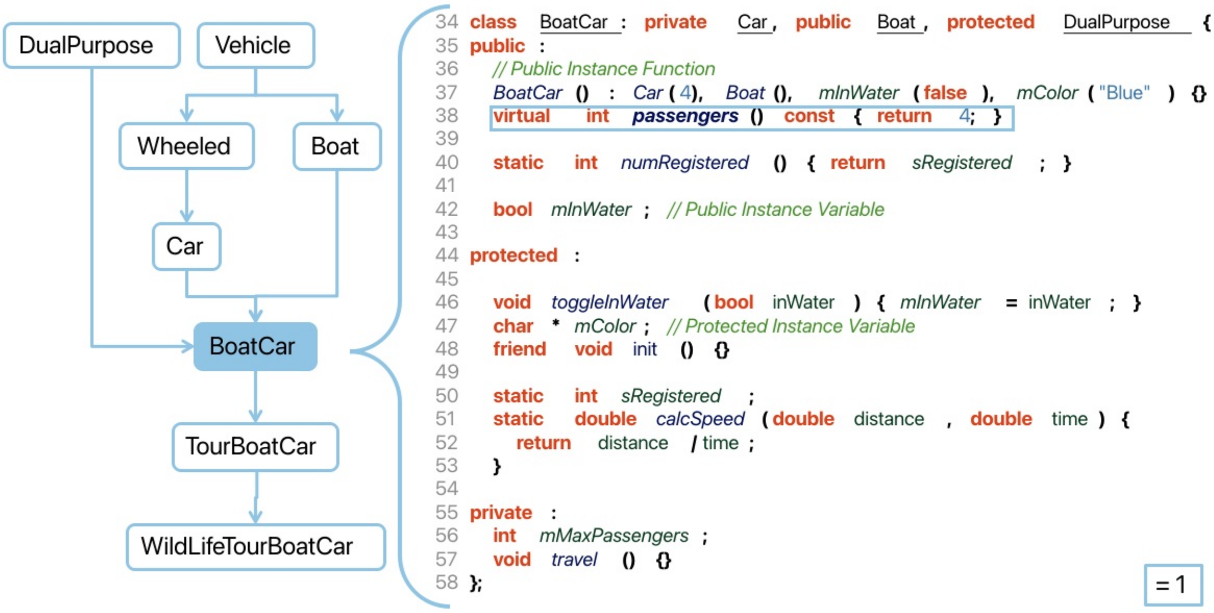
- **Basic:** Class,Struct
- **C/C++:** Class,Struct,Union
- **C#:** Class,Struct
- **Java:** Class,Interface
- **Pascal:** Class,Interface
- **Python:** Class
- **Web:** PHP Class,PHP Interface



Const Methods

API Name: CountDeclMethodConst
Description: Number of local const methods.
Available For:

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



```
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    ()   {}
};
```

Default Methods

API Name: CountDeclMethodDefault
Description: Number of local default methods.
Available For:

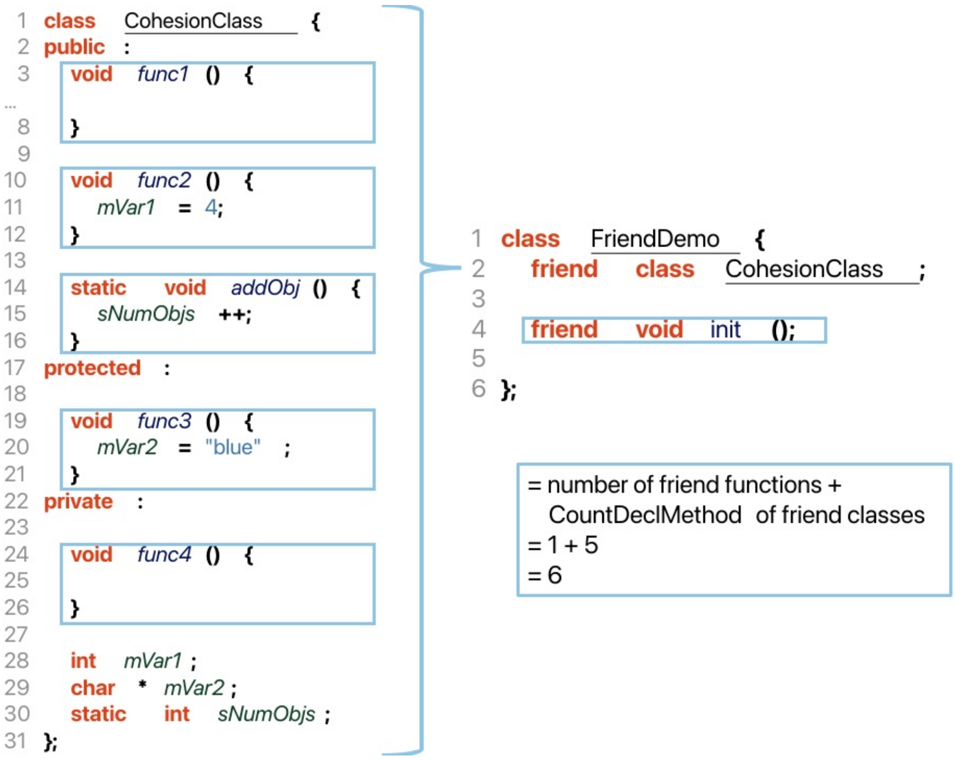
- **Java:** Project,File,Class,Interface

Friend Methods

API Name: CountDeclMethodFriend
Research Name: Lorenz & Kidd - Number of Friends (NF), Number of Friend Methods (NFM)
Description: Number of local (not inherited) friend methods. [aka NFM (number of friend methods), NF (number of friends)]]

The number of friend functions plus the CountDeclMethod of friend classes.
Available For:

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



```
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

Internal Methods

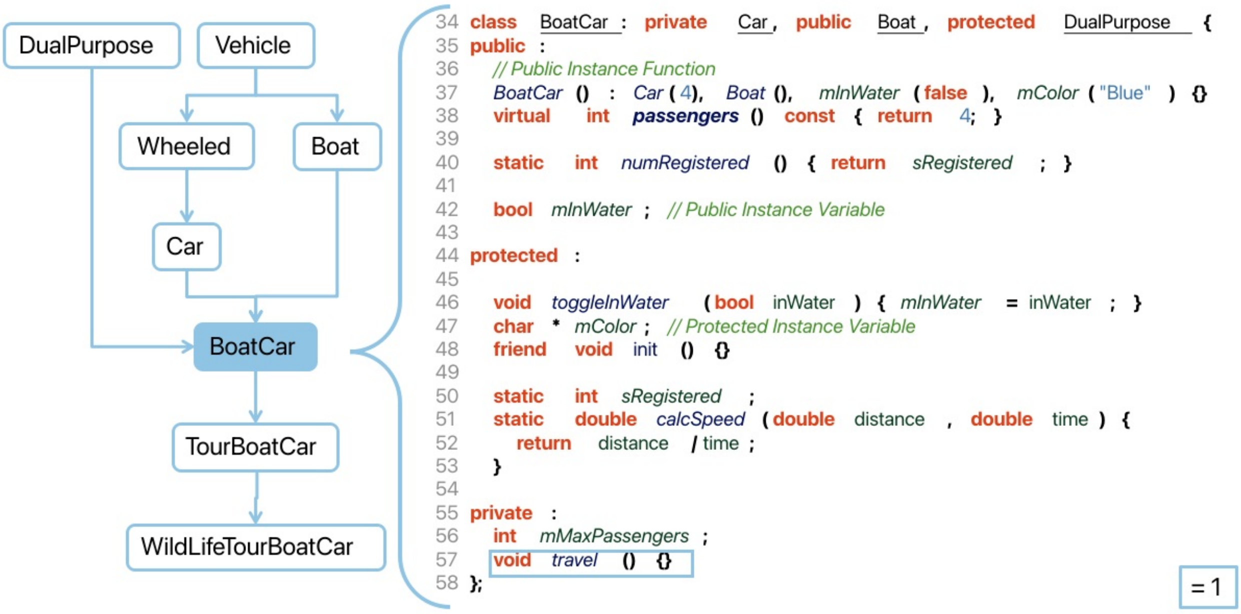
API Name: CountDeclMethodInternal
Description: Number of local internal methods.
Available For:

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

Private Methods

API Name: CountDeclMethodPrivate
Research Name: Number Private Methods (NPRM)
Description: Number of local (not inherited) private methods. [aka NPRM]
Available For:

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

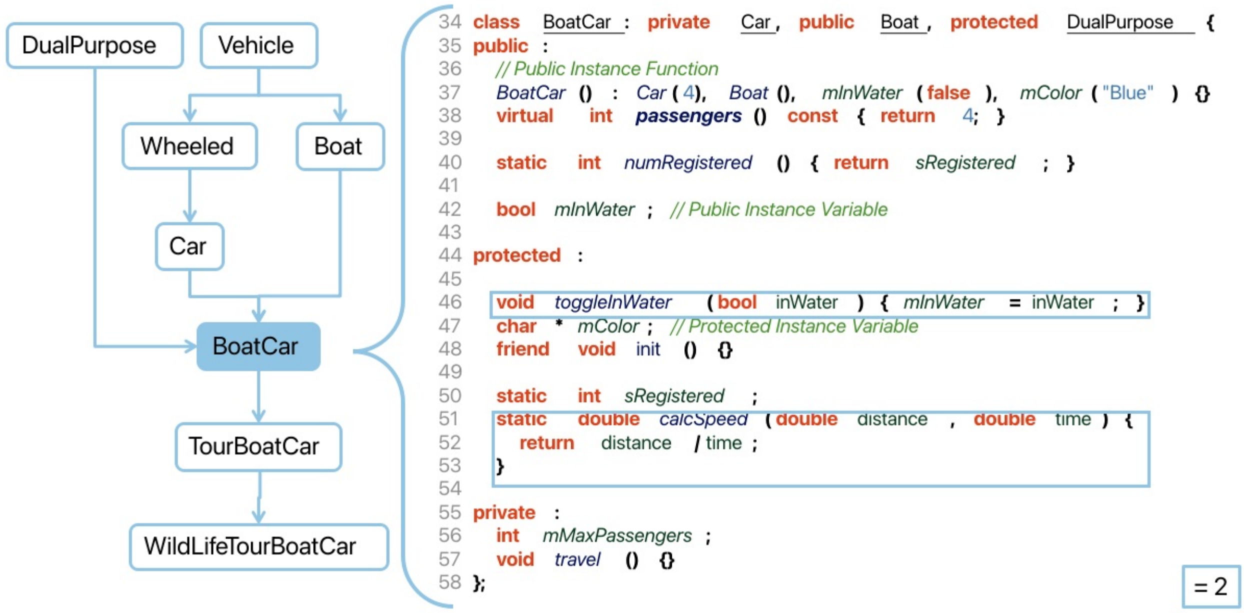


```
class BoatCar : private Car, public Boat, protected DualPurpose {
public :
   // Public Instance Function
   BoatCar () : Car (4), Boat (), mInWater (false ), mColor ("Blue" ) {}
   virtual int passengers () const { return 4; }

   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 () {}
};
```

Protected Methods

API Name: CountDeclMethodProtected
Description: Number of local protected methods.
Available For:

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



```
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 () {}
};
```

Protected Internal Methods

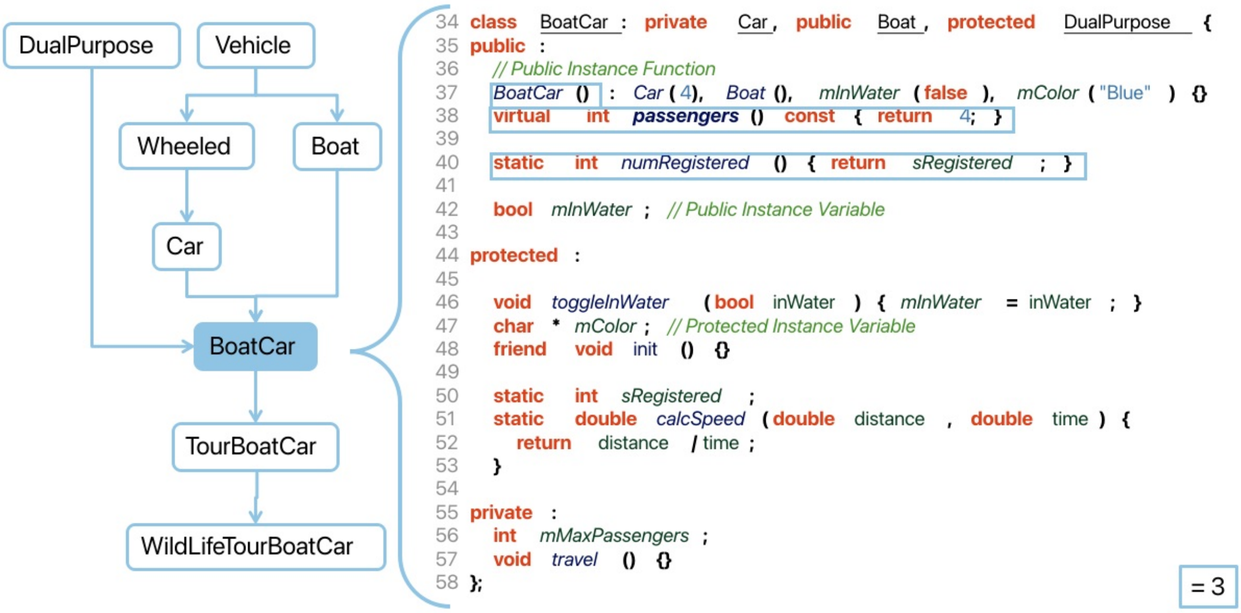
API Name: CountDeclMethodProtectedInternal
Description: Number of local protected internal methods.
Available For:

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

Public Methods

API Name: CountDeclMethodPublic
Research Name: Lorenz & Kidd - Number of Public Methods (PM, NPM)
Description: Number of local (not inherited) public methods. [aka PM, NPM]
Available For:

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



```
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 Private Methods

API Name: CountDeclMethodStrictPrivate
Description: Number of local strict private methods.
Available For:

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

Strict Published Methods

API Name: CountDeclMethodStrictPublished
Description: Number of local strict published methods.
Available For:

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

Modules

API Name: CountDeclModule
Description: Number of modules.
Available For:

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

Program Units

API Name: CountDeclProgUnit
Description: Number of non-nested modules, block data units, and subprograms.
Available For:

- **Fortran:** Project,File

Properties

API Name: CountDeclProperty
Description: Number of properties.
Available For:

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

Auto-Implemented Properties

API Name: CountDeclPropertyAuto
Description: Number of auto-implemented properties.
Available For:

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

Subprograms

API Name: CountDeclSubprogram
Description: Number of subprograms.
Available For:

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

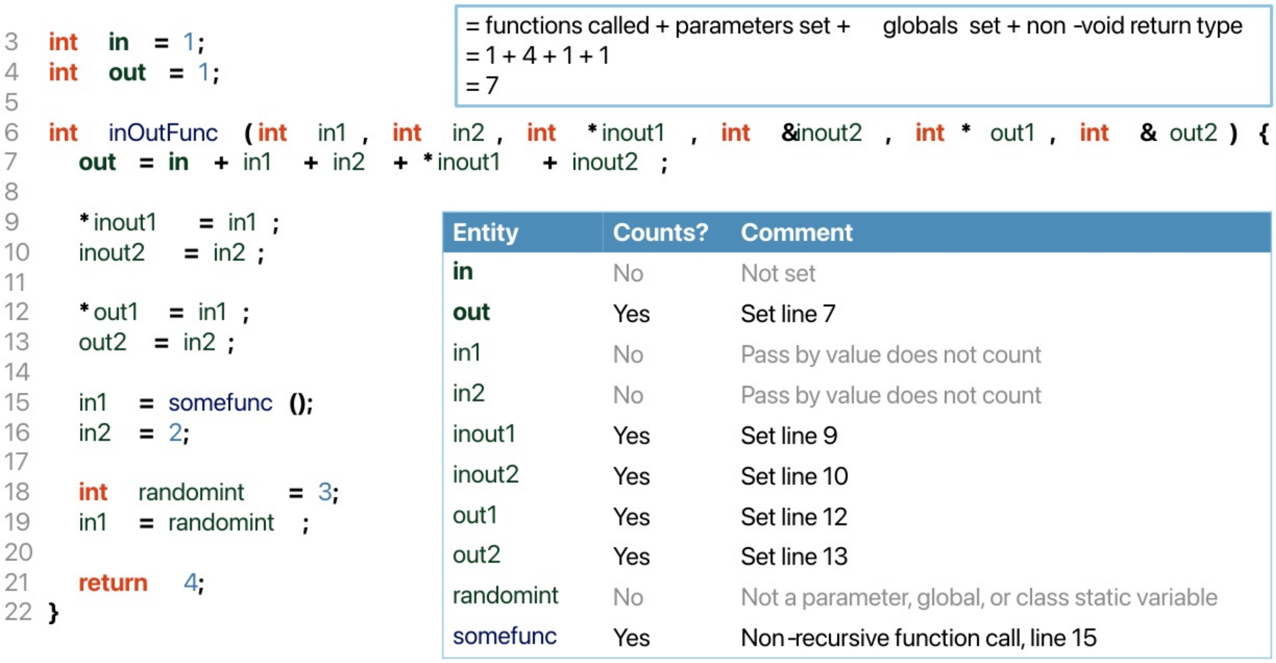
Inputs

API Name: CountInput
Research Name:FANIN (Infomational fan-in)
Description: Number of calling subprograms plus global variables read. [aka FANIN]

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

Available For:

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



```
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  }
```

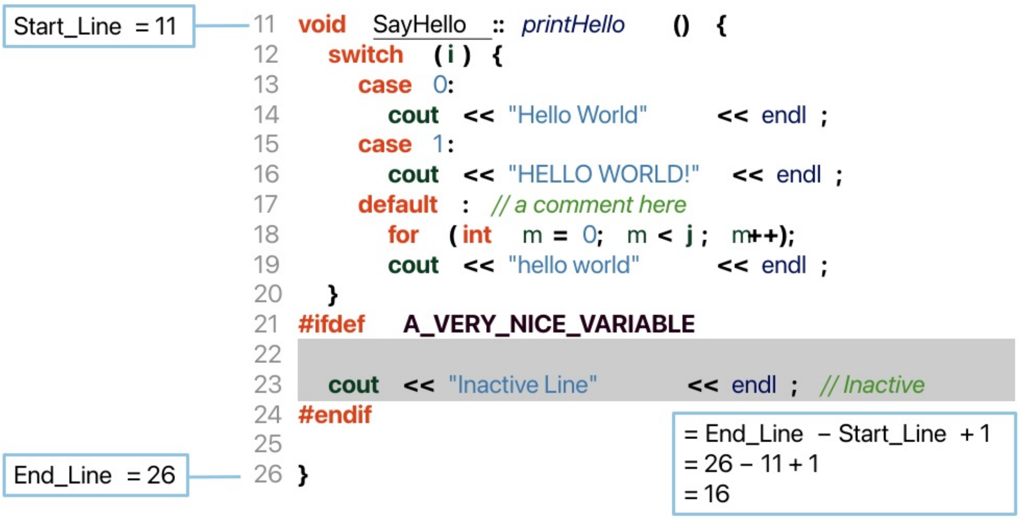
= functions called + parameters set + globals set + non -void return type
= 1 + 4 + 1 + 1
= 7

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

Lines

API Name: CountLine
Research Name: Number of Lines (NL)
Description: Number of physical lines. [aka NL]
Available For:

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



Start_Line = 11

```
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 }
```

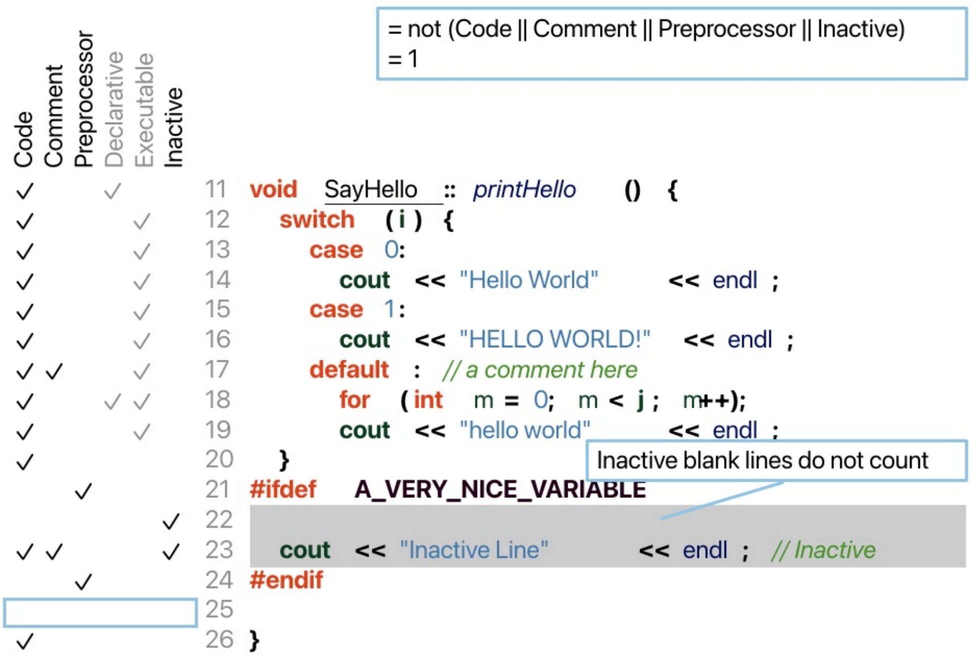
End_Line = 26

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

Blank Lines

API Name: CountLineBlank
Research Name: Blank Lines of Code (BLOC)
Description: Number of blank lines. [aka BLOC (blank lines of code)]
Available For:

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



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

```
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

}
```

Inactive blank lines do not count

Blank Lines (HTML)

API Name: CountLineBlankHtml
Description: Number of blank HTML lines.
Available For:

- **Web:** Project,File

Blank Lines (JavaScript)

API Name: CountLineBlankJavascript
Description: Number of blank JavaScript lines.
Available For:

- **Web:** Project,File

Blank Lines (PHP)

API Name: CountLineBlankPhp
Description: Number of blank PHP lines.
Available For:

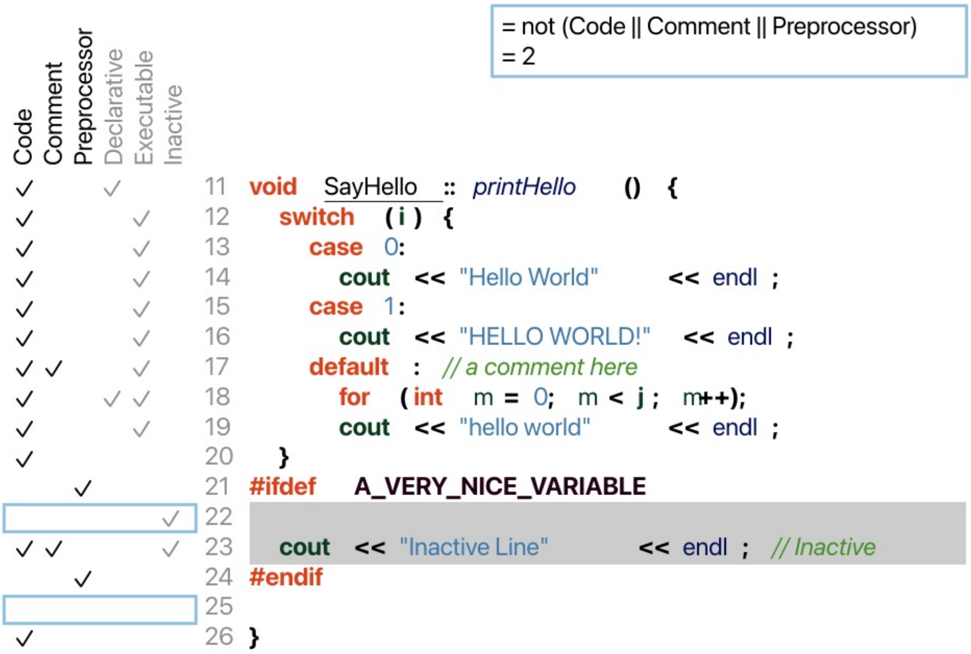
- **Web:** Project,File,PHP Class,PHP Interface

Blank Lines (Includes Inactive)

API Name: CountLineBlankWithInactive

Description: Number of blank lines, including inactive regions.
Available For:

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



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

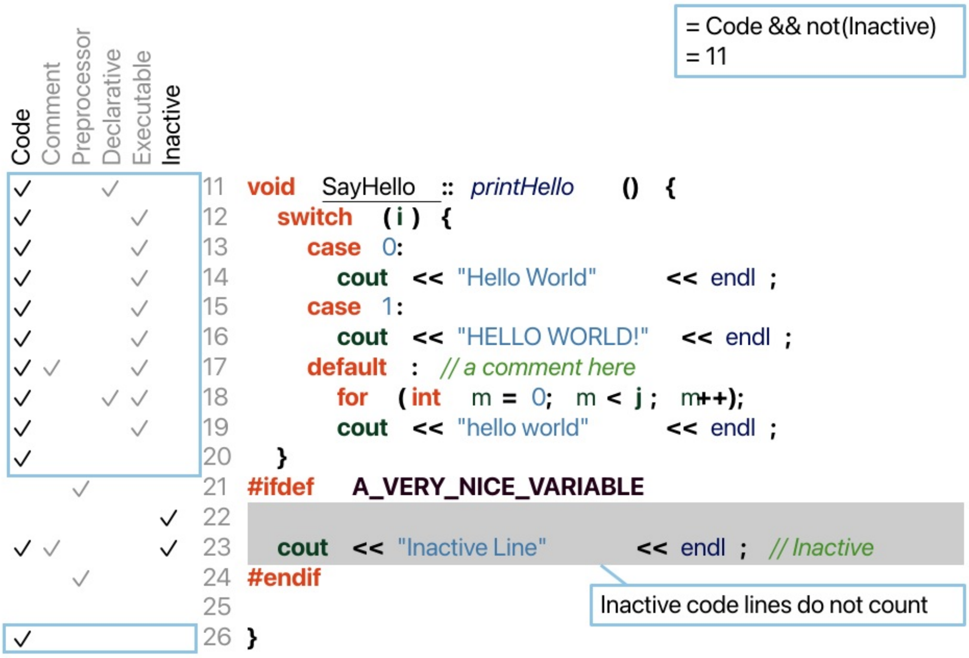
Code	Comment	Preprocessor	Declarative	Executable	Inactive		
✓			✓			11	`void SayHello :: printHello () {`
✓				✓		12	`  switch (i) {`
✓				✓		13	`    case 0:`
✓				✓		14	`      cout << "Hello World" << endl ;`
✓				✓		15	`    case 1:`
✓				✓		16	`      cout << "HELLO WORLD!" << endl ;`
✓	✓			✓		17	`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

API Name: CountLineCode
Research Name: Lines of Code (LOC), Source Lines of Code (SLOC)
Description: Number of lines containing source code. [aka LOC, SLOC]

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

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



```
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

}
```

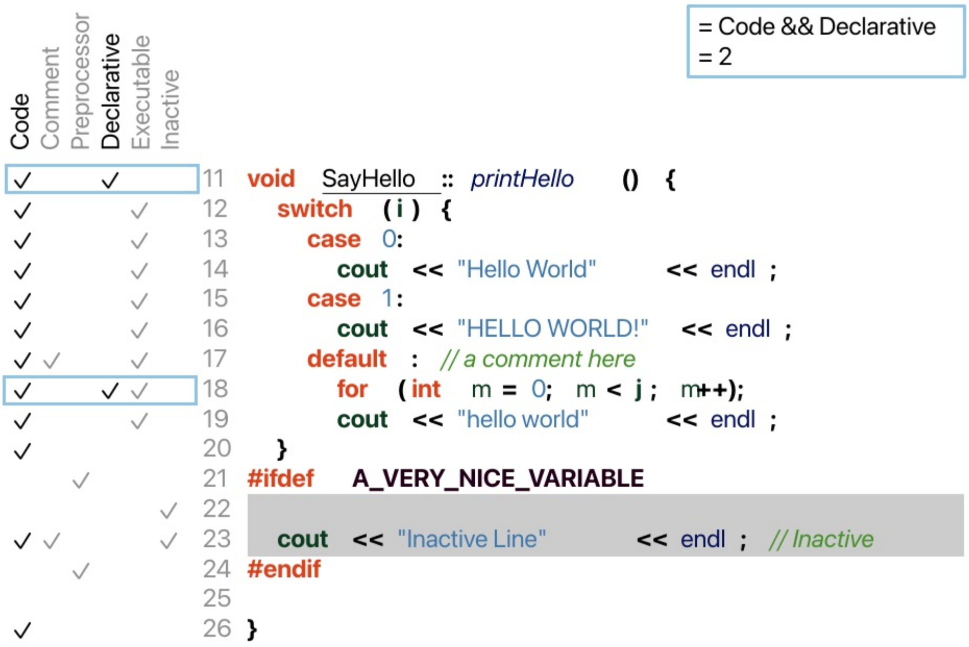
= Code && not(Inactive)
= 11

Inactive code lines do not count

Declarative Code Lines

API Name: CountLineCodeDecl
Description: Number of lines containing declarative source code. Note that a line can be declarative and executable - "int i =0;" for instance.
Available For:

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

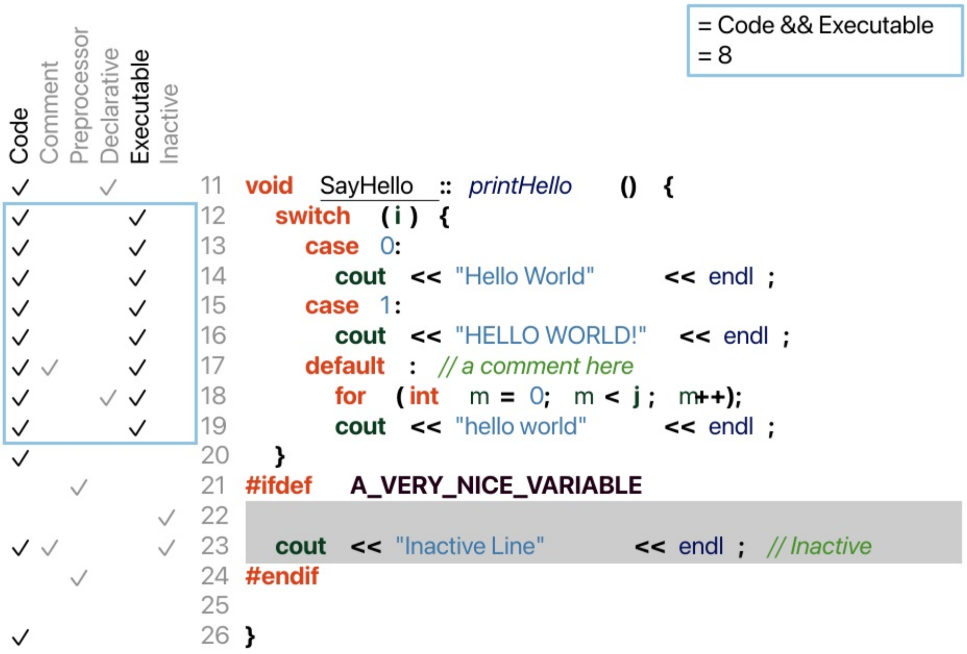


```
void SayHello :: printHello () {
  switch (i) {
    case 0:
      cout << "Hello World" << endl ;
    case 1:
      cout << "HELLO WORLD!" << endl ;
    default : // a comment here
      for (int m = 0; m < j; m++);
      cout << "hello world" << endl ;
  }
#ifdef A_VERY_NICE_VARIABLE

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

}
```

Executable Code Lines

API Name: CountLineCodeExe
Description: Number of lines containing executable source code.
Available For:

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



```
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

}
```

Code Lines (JavaScript)

API Name: CountLineCodeJavascript
Description: Number of JavaScript lines containing source code.
Available For:

- **Web:** Project,File

Code Lines (PHP)

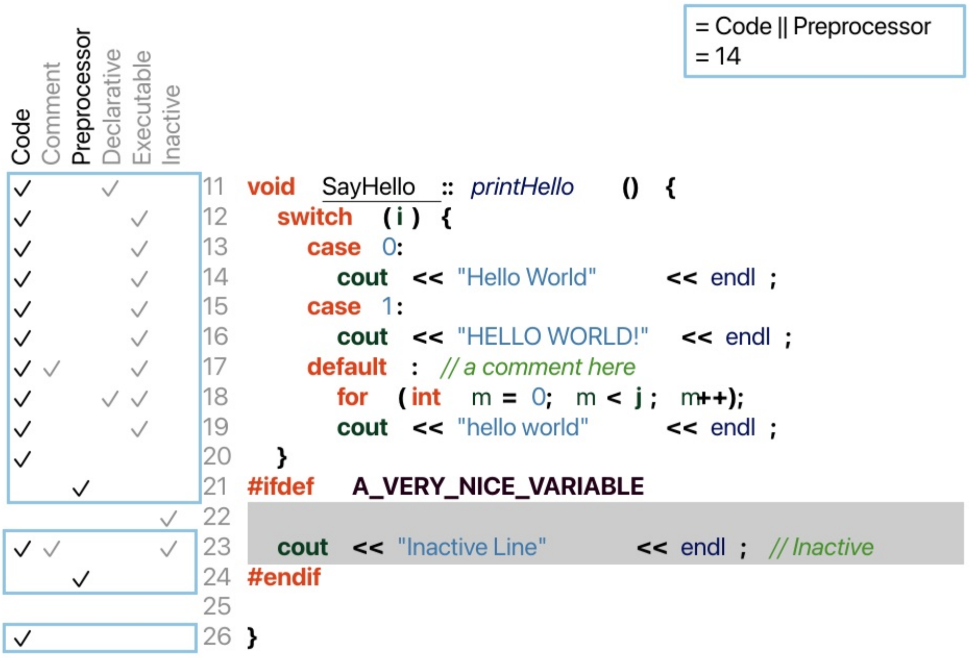
API Name: CountLineCodePhp
Description: Number of PHP lines containing source code.
Available For:

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

Code Lines (Includes Inactive)

API Name: CountLineCodeWithInactive
Description: Number of lines containing source code, including inactive regions.
Available For:

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



```
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

}
```

Comment Lines

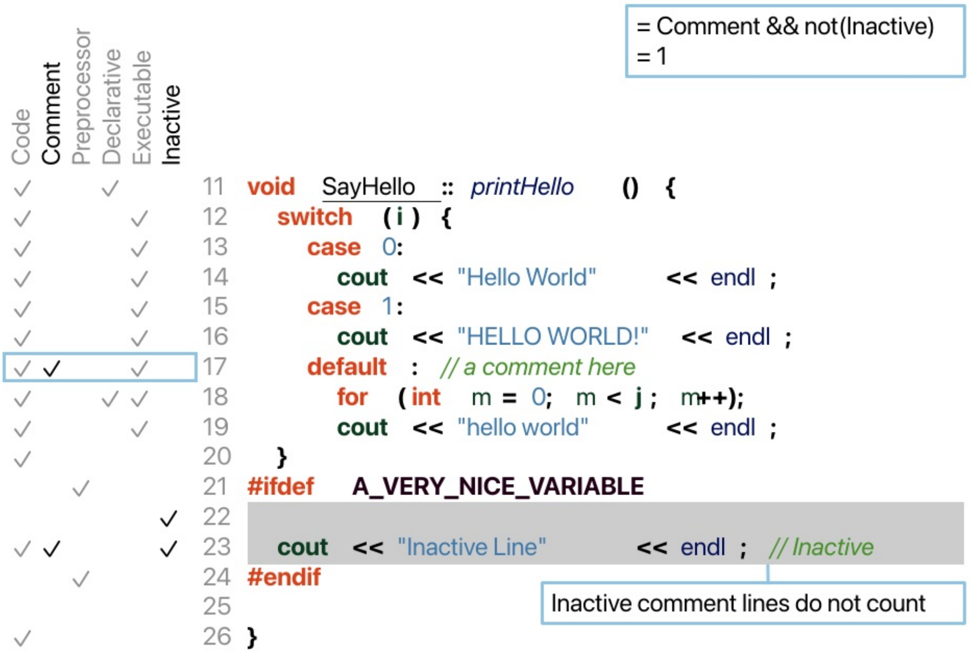
API Name: CountLineComment
Research Name: Comment Lines of Code (CLOC)
Description: Number of lines containing comment. [aka CLOC]

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

int j = 1; // comment

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

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



= Comment && not(Inactive)
= 1

Code	Comment	Preprocessor	Declarative	Executable	Inactive	Line	Source
✓			✓			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 Lines (HTML)

API Name: CountLineCommentHtml
Description: Number of HTML lines containing comment.
Available For:

- **Web:** Project,File

Comment Lines (JavaScript)

API Name: CountLineCommentJavascript
Description: Number of JavaScript lines containing comment.
Available For:

- **Web:** Project,File

Comment Lines (PHP)

API Name: CountLineCommentPhp
Description: Number of PHP lines containing comment.
Available For:

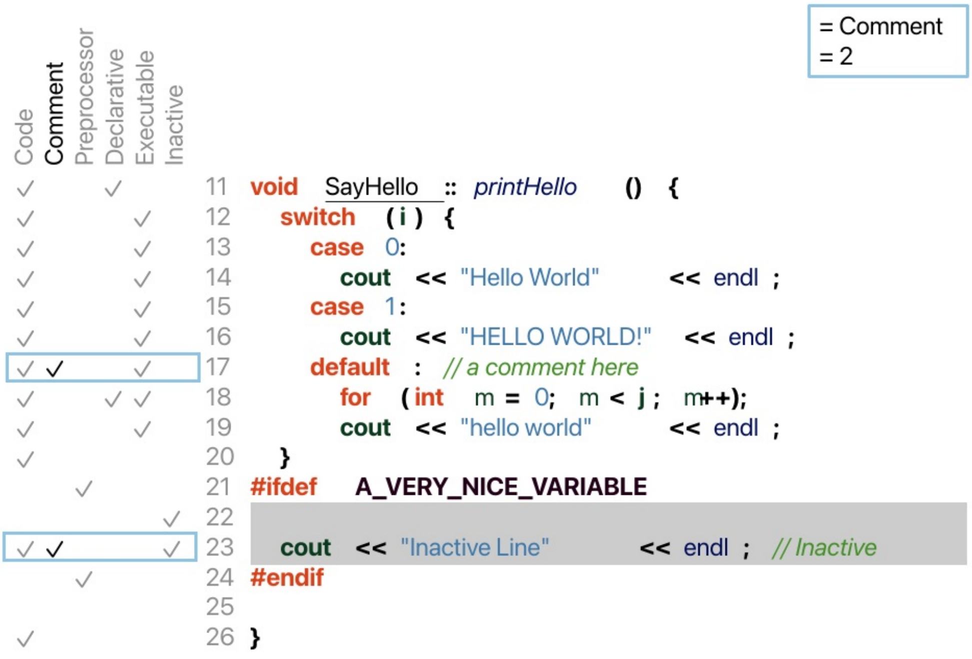
- **Web:** Project,File,PHP Class,PHP Interface

Comment Lines (Includes Inactive)

API Name: CountLineCommentWithInactive

Description: Number of lines containing comment, including inactive regions.
Available For:

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



```
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

}
```

Lines (HTML)

API Name: CountLineHtml
Description: Number of all HTML lines.
Available For:

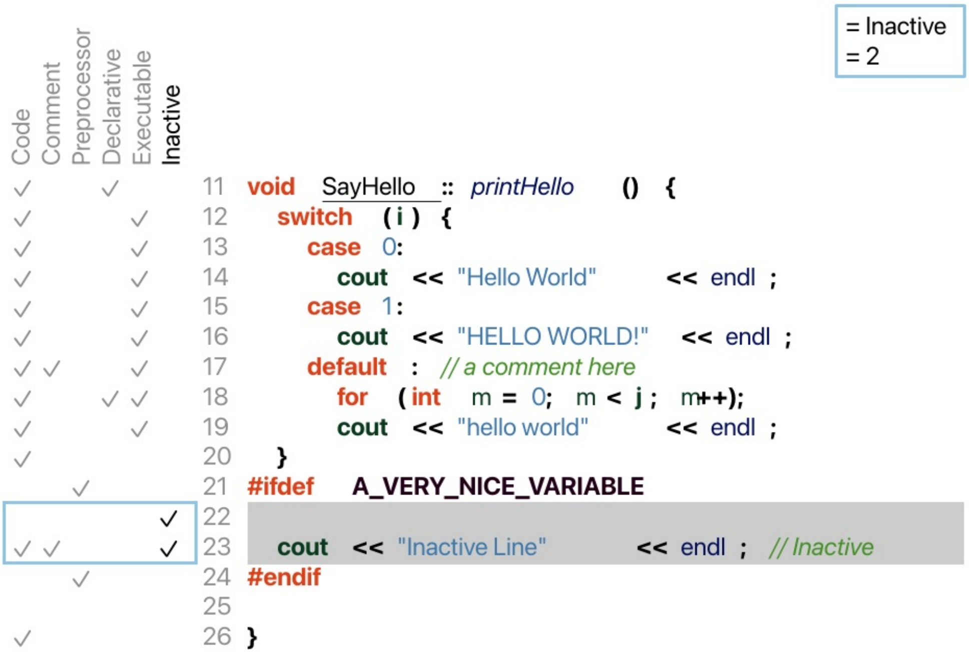
- **Web:** Project,File

Inactive Lines

API Name: CountLineInactive
Description: Number of inactive lines.

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

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



= 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	`}`

Lines (JavaScript)

API Name: CountLineJavascript
Description: Number of all JavaScript lines.
Available For:

- **Web:** Project,File

Lines (PHP)

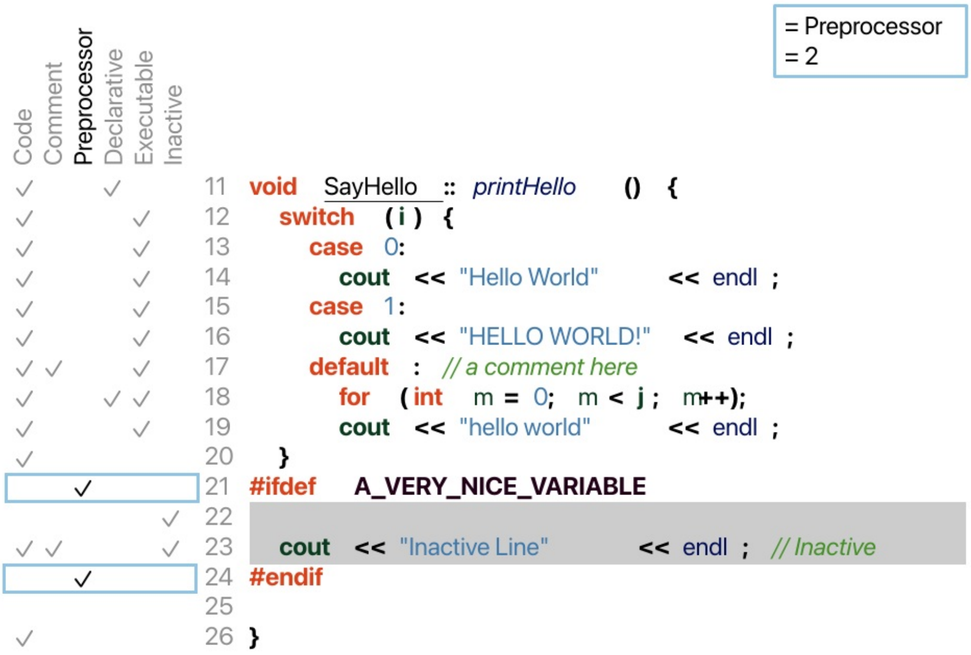
API Name: CountLinePhp
Description: Number of all PHP lines.
Available For:

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

Preprocessor Lines

API Name: CountLinePreprocessor
Description: Number of preprocessor lines.
Available For:

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



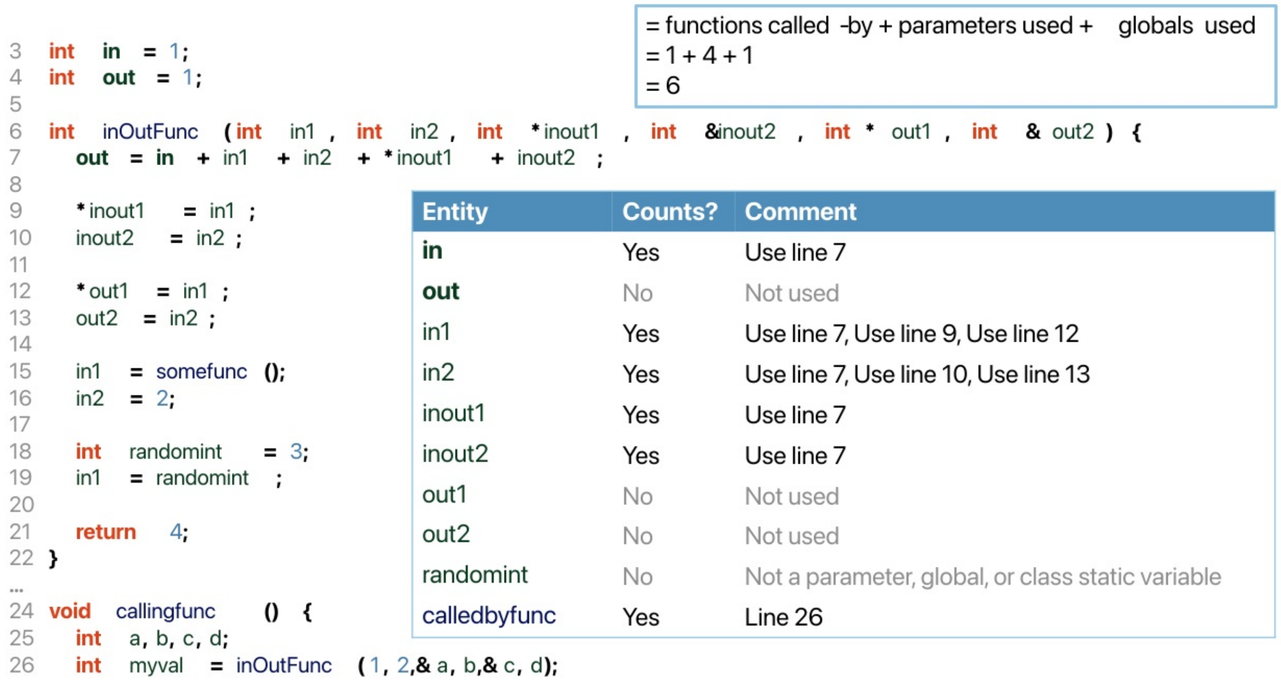
Code	Comment	Preprocessor	Declarative	Executable	Inactive		
✓			✓			11	`void SayHello :: printHello () {`
✓				✓		12	`switch (i) {`
✓				✓		13	`case 0:`
✓				✓		14	`cout << "Hello World" << endl ;`
✓				✓		15	`case 1:`
✓				✓		16	`cout << "HELLO WORLD!" << endl ;`
✓	✓			✓		17	`default : // a comment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

Outputs

API Name: CountOutput
Research Name:FANOUT (Infomational fan-out)
Description: Number of called subprograms plus global variables set. [aka FANOUT]

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

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



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

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

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

Coupled Packages

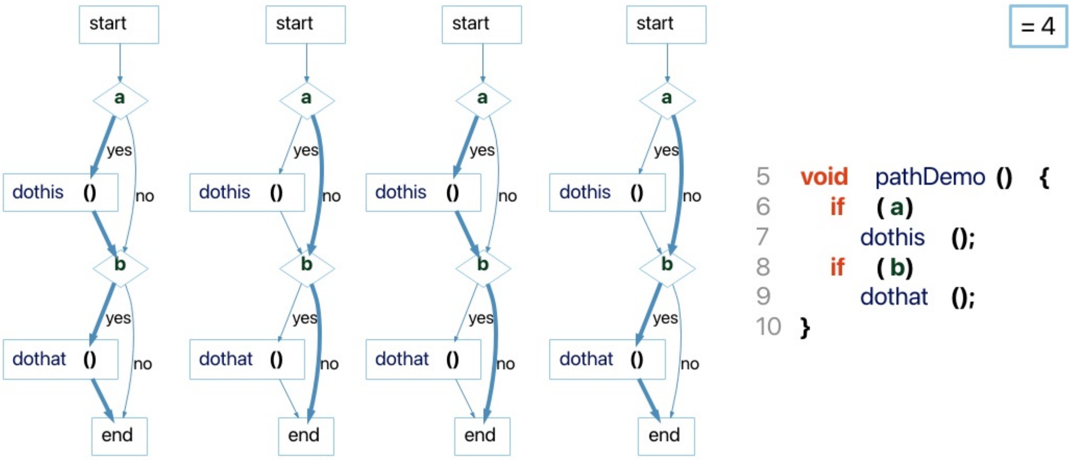
API Name: CountPackageCoupled
Description: Number of other packages coupled to.
Available For:

- **Ada:** Package

Paths

API Name: CountPath
Research Name:NPATH
Description: Number of unique paths trhough a body of code, not counting abnormal exits or gotos. [aka NPATH]
Available For:

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

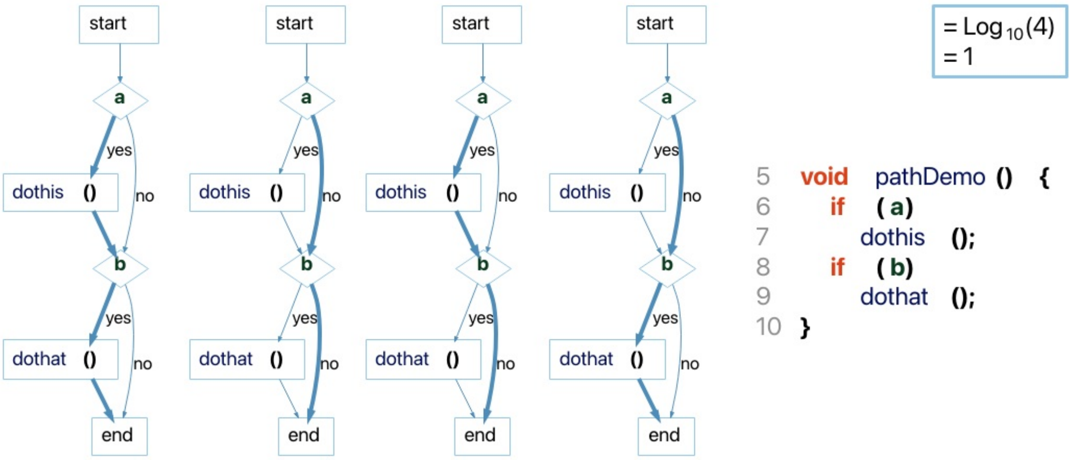


```
void  pathDemo ()  {
   if  ( a)
      dothis  ();
   if  ( b)
      dothat  ();
}
```

Paths Log(x)

API Name: CountPathLog
Description: The base 10 logarithm Log(x) of the number of unique paths though a body of code, not counting abnormal exits or gotos, truncated to an integer value.
Available For:

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



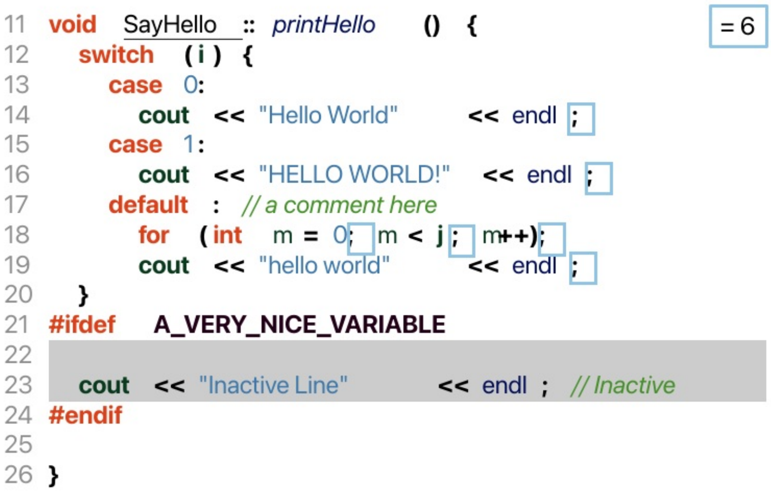
$= \text{Log}_{10}(4)$
$= 1$

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

Semicolons

API Name: CountSemicolon
Description: Number of semicolons.
Available For:

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

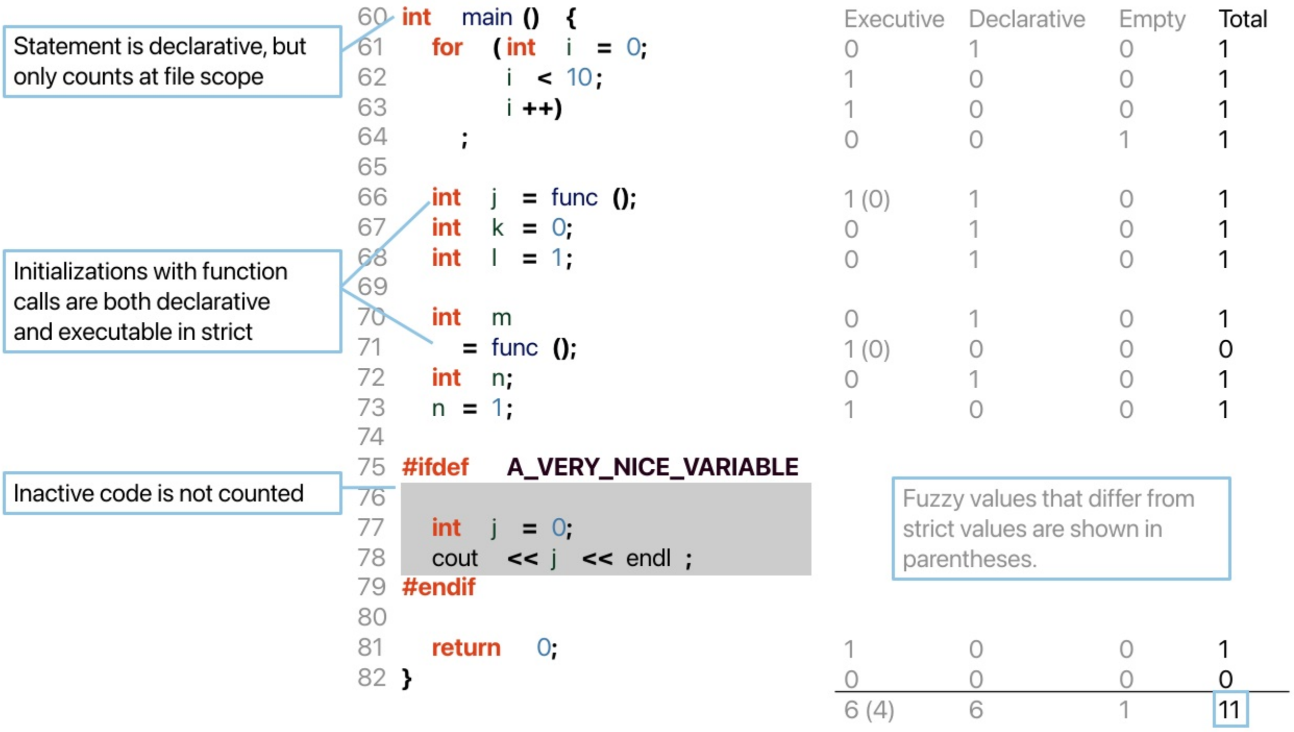


```
void  SayHello  :: printHello    ()  {
   switch   ( i )  {
      case  0:
         cout   <<  "Hello World"       <<  endl ;
      case  1:
         cout   <<  "HELLO WORLD!"     <<  endl ;
      default   :  // a comment here
         for   ( int    m  =  0;  m  <  j ;  m++);
         cout   <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

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

}
```

Statements

API Name: CountStmt
Description: Number of statements.
Available For:

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



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

Statement is declarative, but only counts at file scope

Initializations with function calls are both declarative and executable in strict

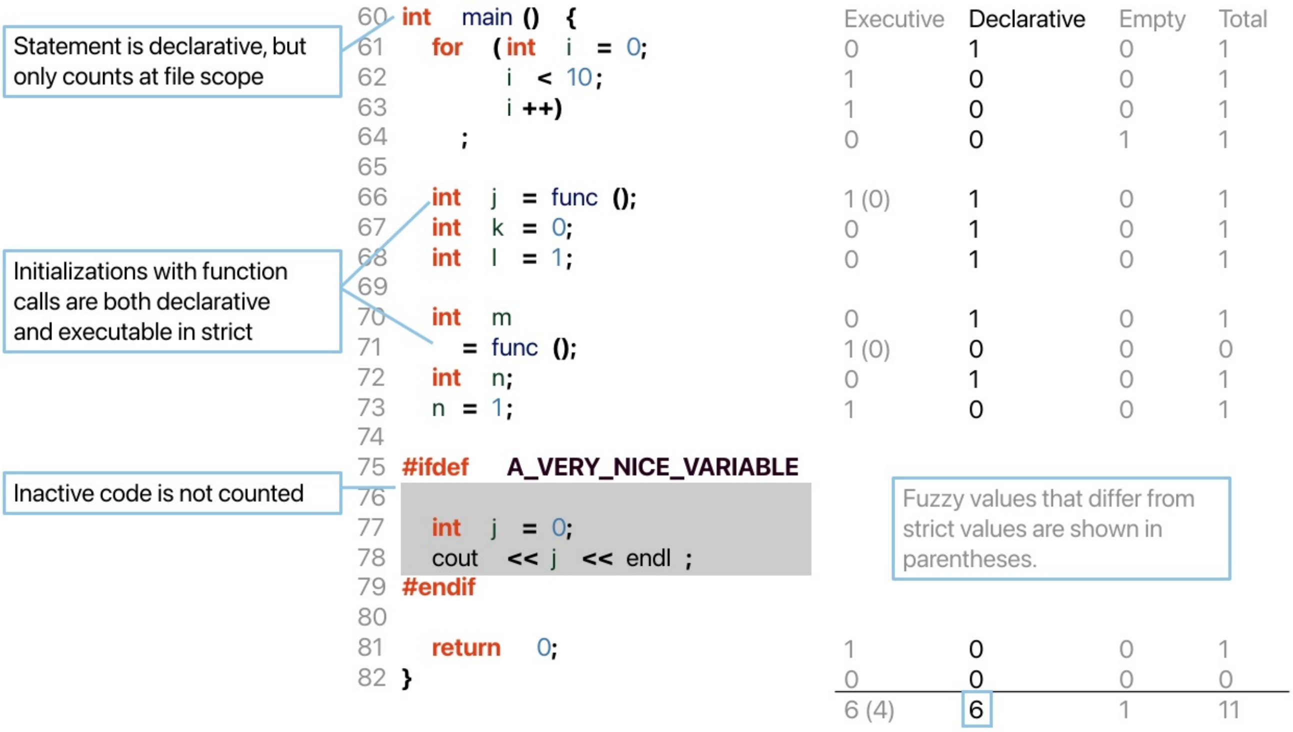
Inactive code is not counted

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

Declarative Statements

API Name: CountStmtDecl
Description: Number of declarative statements.
Available For:

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



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

Declarative Statements (Javascript)

API Name: CountStmtDeclJavascript
Description: Number of JavaScript declarative statements.
Available For:

- **Web:** Project,File

Declarative Statements (PHP)

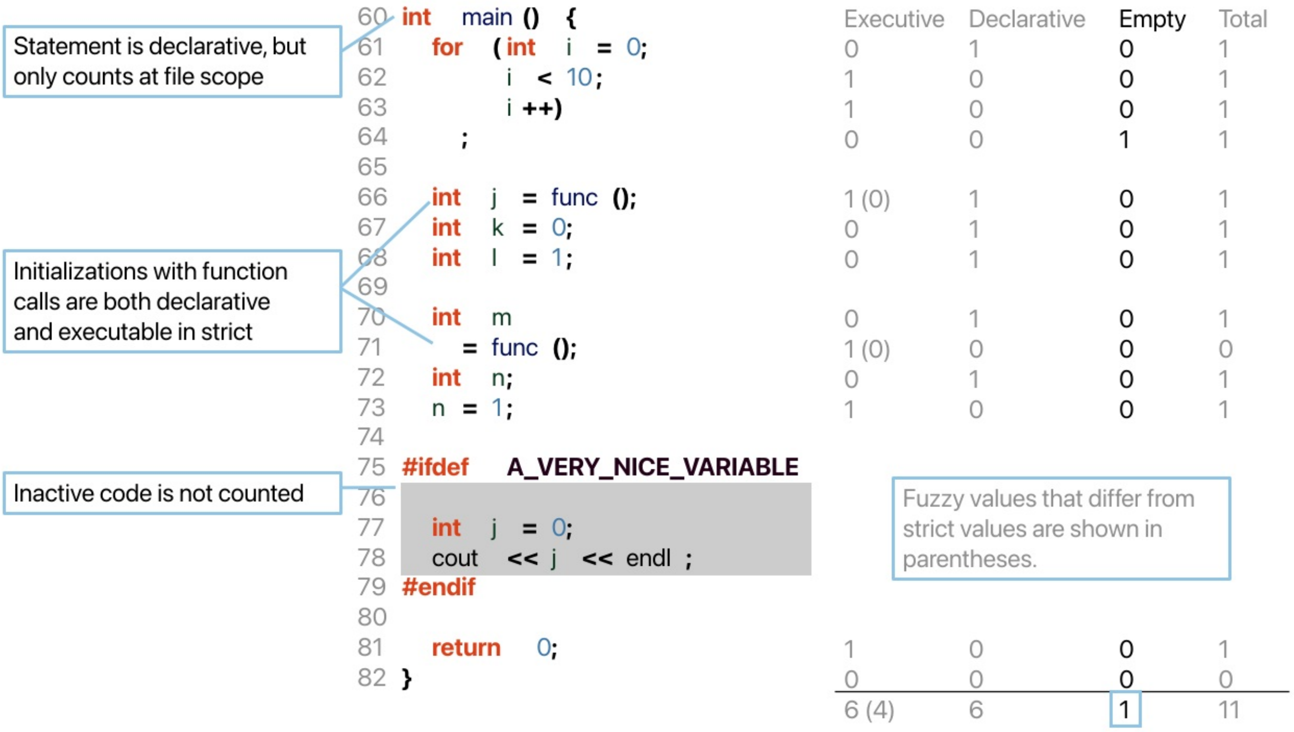
API Name: CountStmtDeclPhp
Description: Number of PHP declarative statements.
Available For:

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

Empty Statements

API Name: CountStmtEmpty
Description: Number of empty statements.
Available For:

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



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.

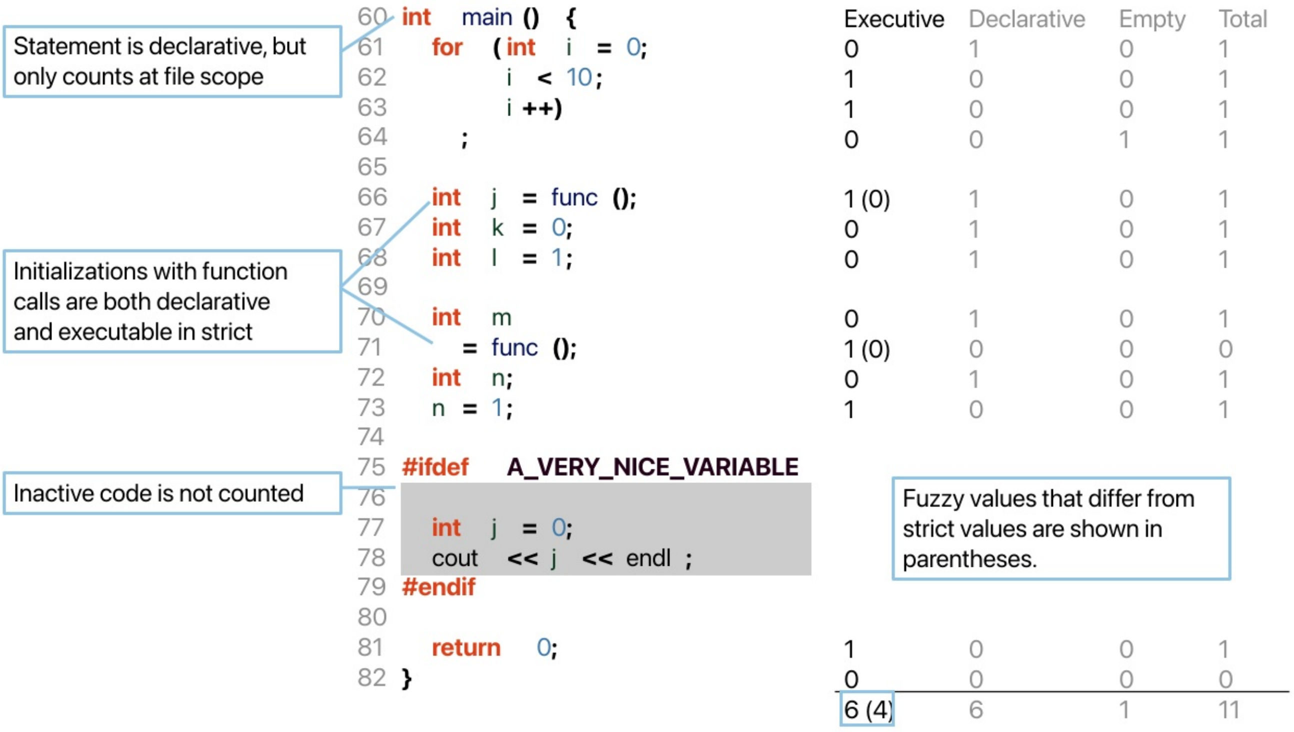
```
int   main ()   {
   for   ( int   i   =  0;
             i   <  10;
             i ++)
       ;

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

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

#ifdef    A_VERY_NICE_VARIABLE

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

   return    0;
}
```

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
0	0	0	0
6 (4)	6	1	11

Executable Statements

API Name: CountStmtExe
Description: Number of executable statements.
Available For:

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



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.

```
int  main ()  {
  for  (int  i  = 0;
          i  < 10;
          i ++)
      ;

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

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

#ifdef   A_VERY_NICE_VARIABLE

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

  return   0;
}
```

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
0	0	0	0
6 (4)	6	1	11

Executable Statements (JavaScript)

API Name: CountStmtExeJavascript
Description: Number of JavaScript executable statements.
Available For:

- **Web:** Project,File

Executable Statements (PHP)

API Name: CountStmtExePhp
Description: Number of PHP executable statements.
Available For:

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

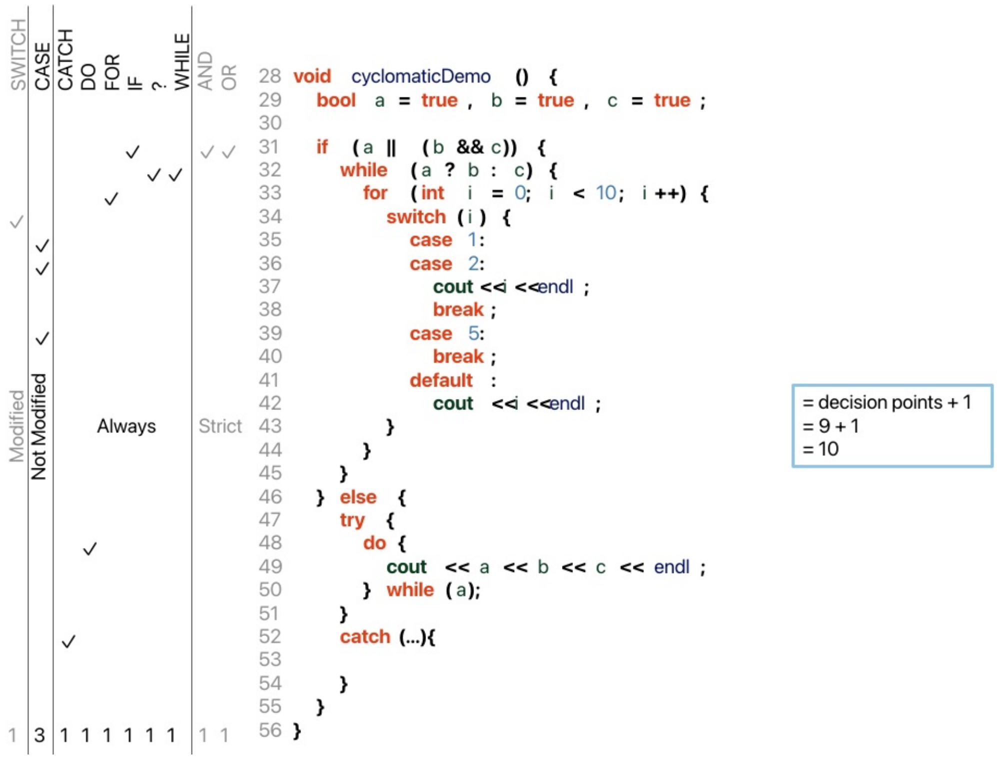
Cyclomatic Complexity

API Name: Cyclomatic
Research Name: McCabe - McCabe Cyclomatic Complexity, CC
Description: McCabe Cyclomatic Complexity, the number of decision points + 1. [aka CC]

McCabe Cyclomatic complexity as per the original NIST paper on the subject. The cyclomatic complexity of any structured program with only one entrance point and one exit point is equal to the number of decision points contained in that program plus one. Understand counts the keywords for decision points (FOR, WHILE, etc)

and then adds 1. For a switch statement, each 'case' is counted as 1. For languages with macros, the expanded macro text is also included in the calculation.
Available For:

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



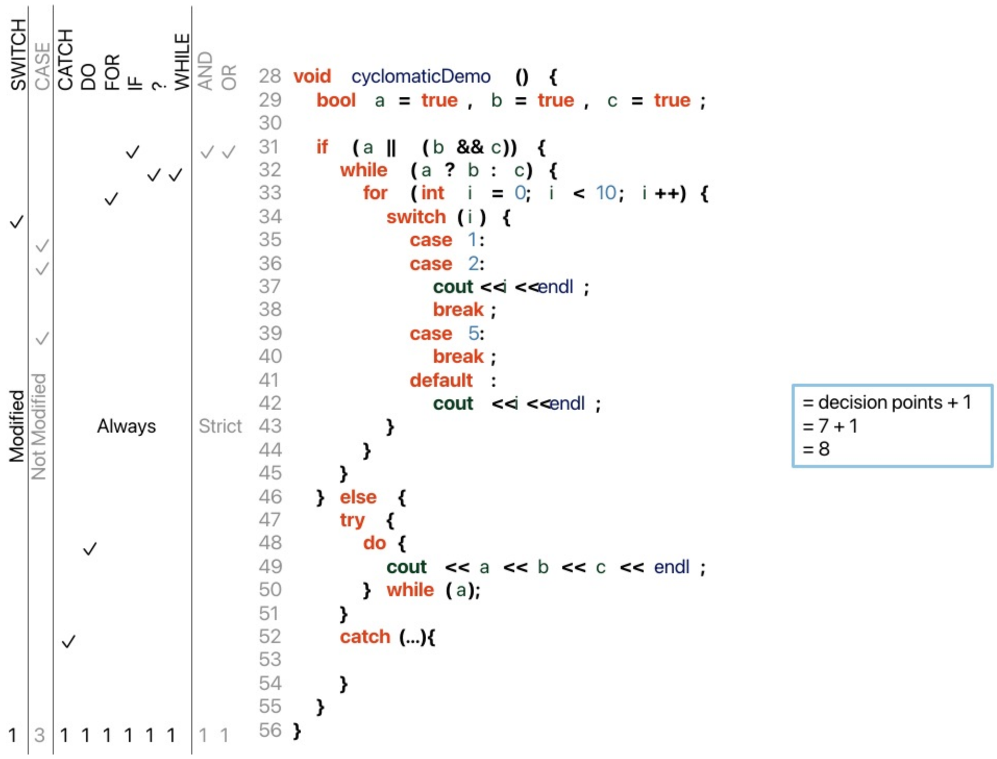
```
void  cyclomaticDemo  ()  {
  bool  a = true ,  b = true ,  c = true ;

  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 (...){

    }
  }
}
```

Modified Cyclomatic Complexity

API Name: CyclomaticModified
Research Name: McCabe - McCabe Modified Cyclomatic Complexity, CC3
Description: Modified McCabe Cyclomatic complexity [aka CC3].

The Cyclomatic Complexity except that each decision in a multi-decision structure (switch in C/Java, Case in Ada, computed Goto and arithmetic if in FORTRAN) statement is not counted and instead the entire multi-way decision structure counts as 1.
Available For:

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



```
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

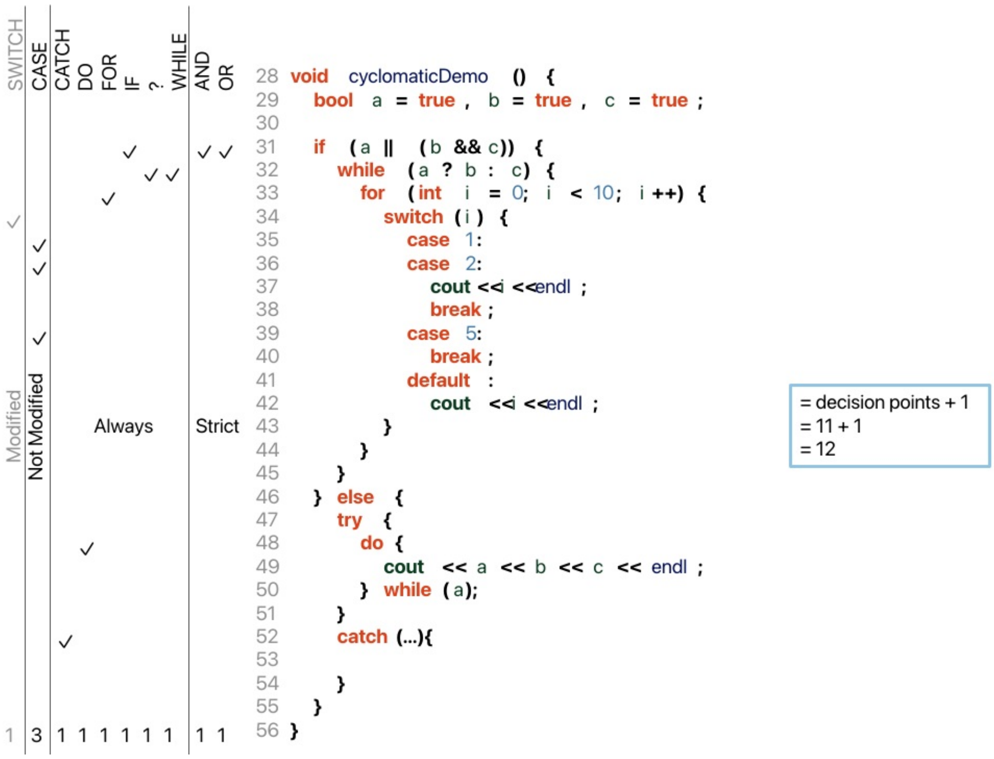
Strict Cyclomatic Complexity

API Name: CyclomaticStrict
Research Name: McCabe - McCabe Strict Cyclomatic Complexity, CC2
Description: Strict McCabe Cyclomatic Complexity [aka CC2].

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

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

- **Fortran:** Module,Block Data,Function,Program,Subroutine
- **Java:** Method
- **Jovial:** Subroutine
- **Pascal:** Compunit,Function,Procedure
- **Python:** File,Function
- **VHDL:** Procedure,Function,Process
- **Web:** File



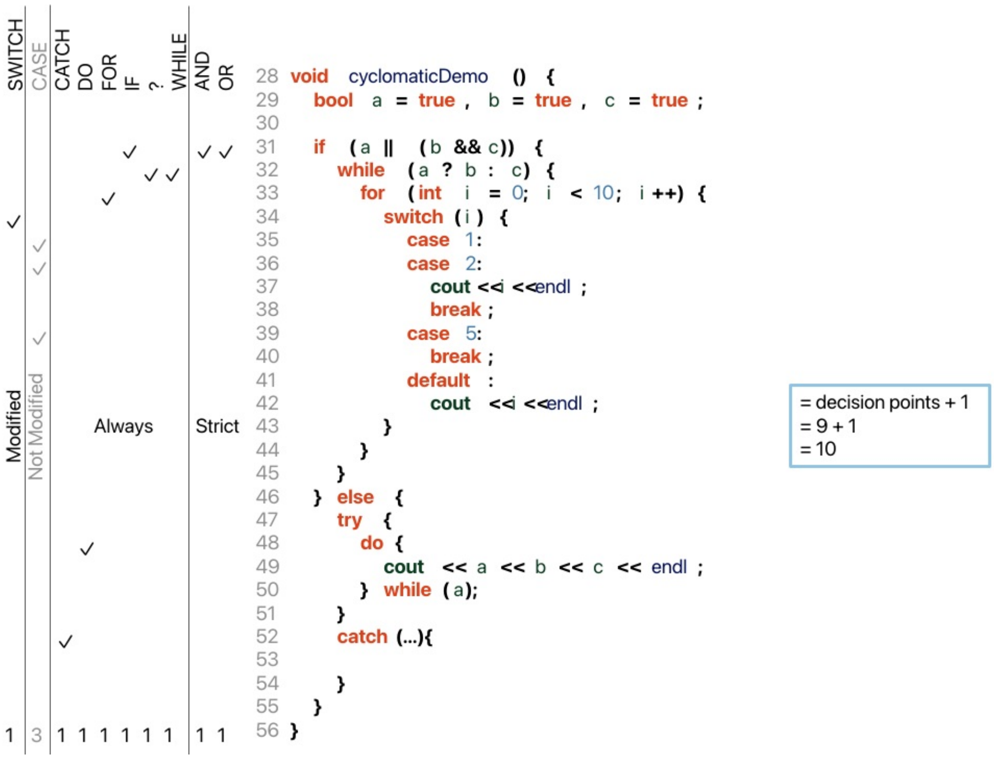
```
void  cyclomaticDemo  ()  {
  bool  a  =  true ,  b  =  true ,  c  =  true ;

  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  (...){

    }
  }
}
```

Strict Modified Cyclomatic Complexity

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

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

- **Web:** File



```
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 (...){

    }
  }
}
```

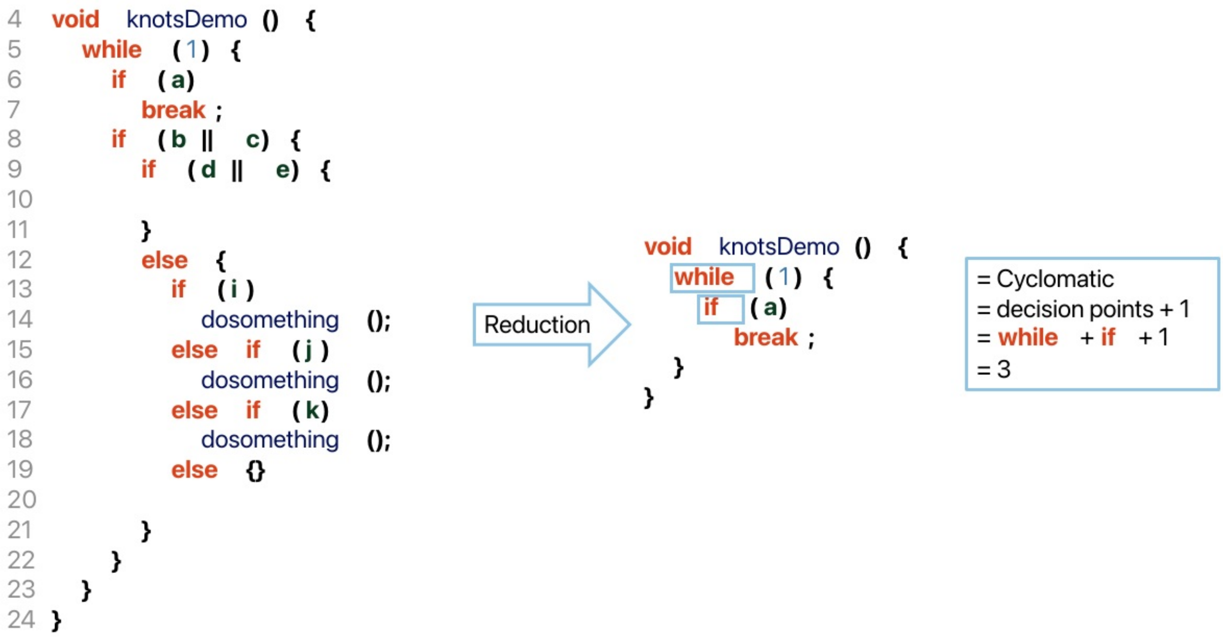
Essential Complexity

API Name: Essential
Research Name:ev(G)
Description: The number of decision points + 1 after control graph reduction. [aka ev(G)]

Essential complexity is the cyclomatic complexity after iteratively replacing all well structured control structures with a single statement. Structures such as if-then-else and while loops are considered well structured. Understand calculates the essential complexity by removing all the structured subgraphs from the control graph and then calculating the complexity. A graph that has only the regular single entry/single exit loops or branches will be reducible to a graph with complexity one. Any branches into or out of a loop or decision will make the graph non-reducible and will have Essential Complexity > 2. (You never get 2 since a graph with complexity 2 is always reducible to a graph with complexity 1)
Available For:

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

- **Python:** File,Function
- **Web:** File



```
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 ;
  }
}
```

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

Strict Modified Essential Complexity

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

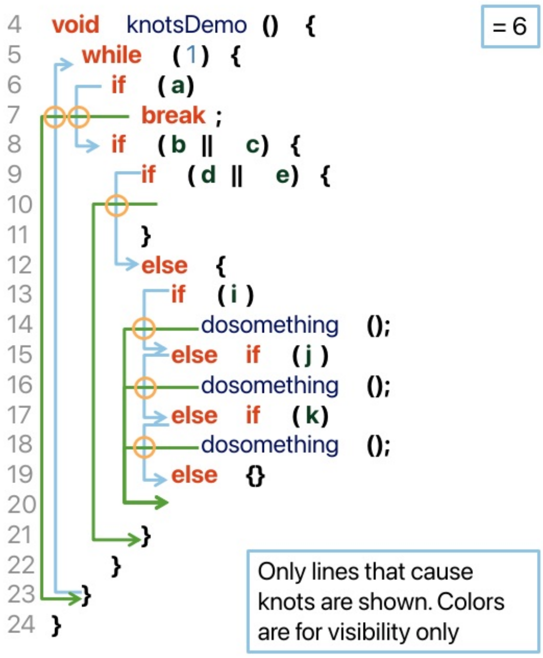
- **Ada:** Type,Entry,Function,Package,Procedure,Protected,Task

Knots

API Name: Knots
Description: Measure of overlapping jumps.

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

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



```
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 {}

      }
    }
  }
}
```

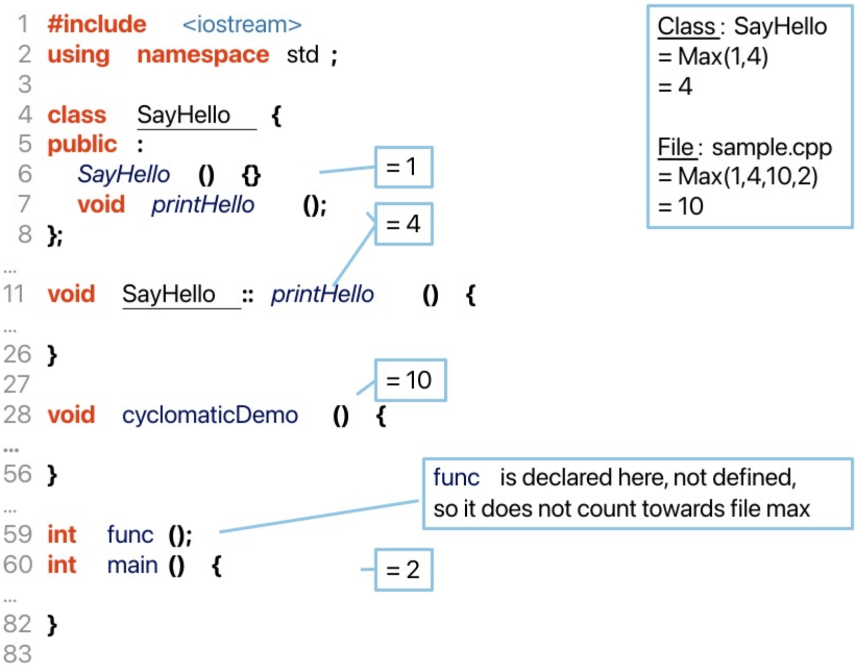
= 6

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

Max Cyclomatic Complexity

API Name: MaxCyclomatic
Description: Maximum cyclomatic complexity of all nested functions or methods.
Available For:

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



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

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

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

```

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

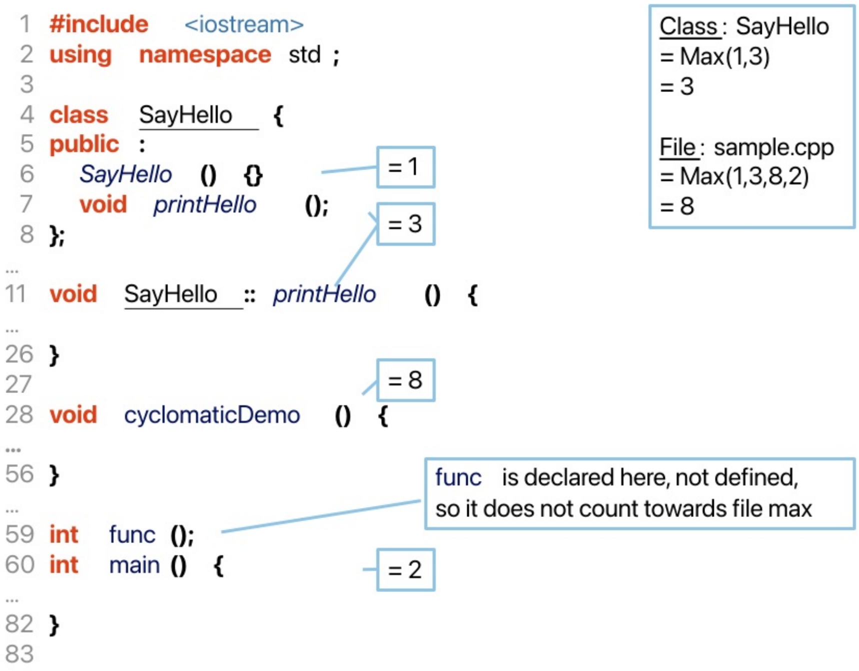
File: sample.cpp
= Max(1,4,10,2)
= 10

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

Max Modified Cyclomatic Compexity

API Name: MaxCyclomaticModified
Description: Maximum modified cyclomatic complexity of nested functions or methods.
Available For:

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



```
#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

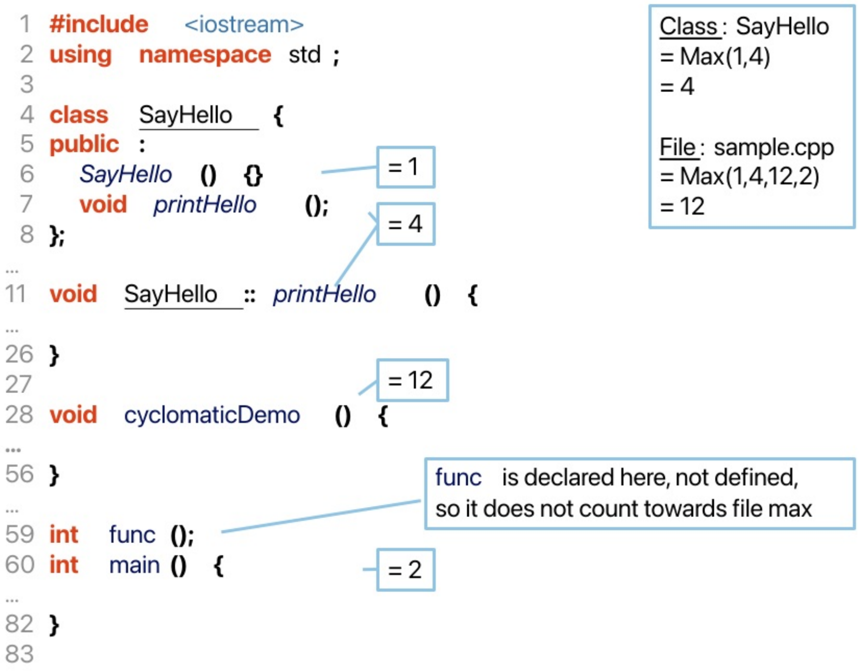
File: sample.cpp
= Max(1,3,8,2)
= 8

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

Max Strict Cyclomatic Complexity

API Name: MaxCyclomaticStrict
Description: Maximum strict cyclomatic complexity of nested functions or methods.
Available For:

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



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

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

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

```

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

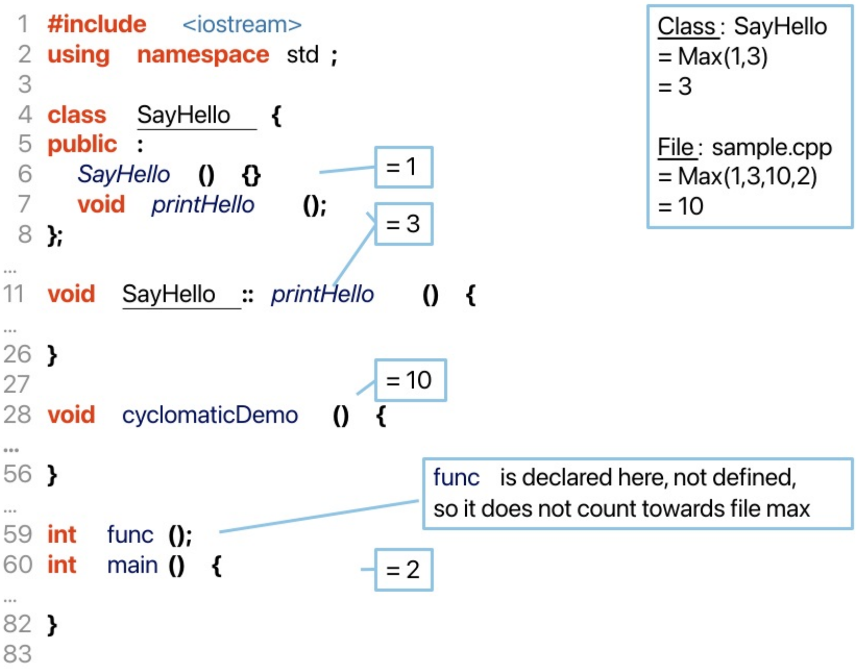
File: sample.cpp
= Max(1,4,12,2)
= 12

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

Max Strict Modified Cyclomatic Complexity

API Name: MaxCyclomaticStrictModified
Description: Maximum strict modified cyclomatic complexity of nested functions or methods.
Available For:

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



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

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

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

```

= 1

= 3

= 10

= 2

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

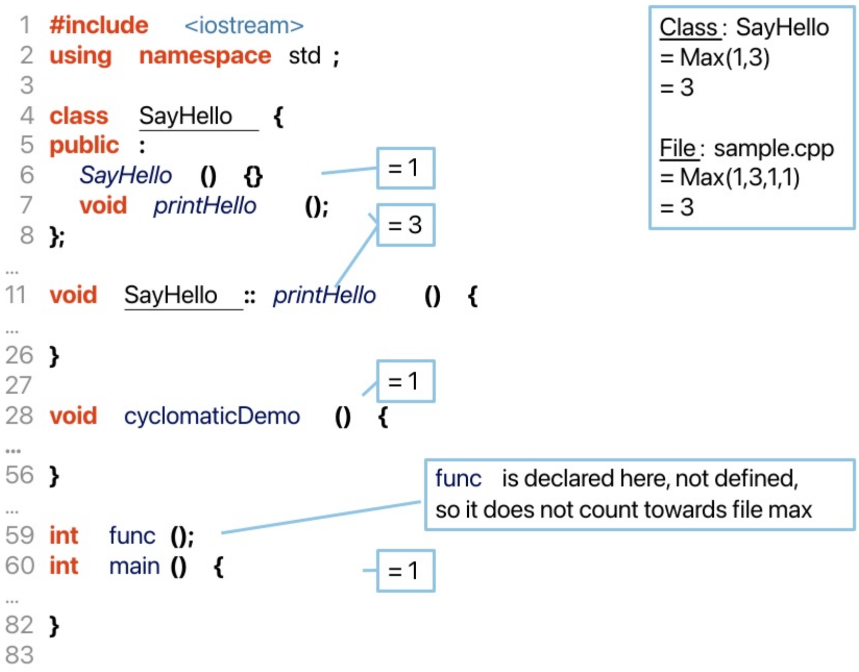
File: sample.cpp
= Max(1,3,10,2)
= 10

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

Max Essential Complexity

API Name: MaxEssential
Description: Maximum essential complexity of all nested functions or methods.
Available For:

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



```
#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 max

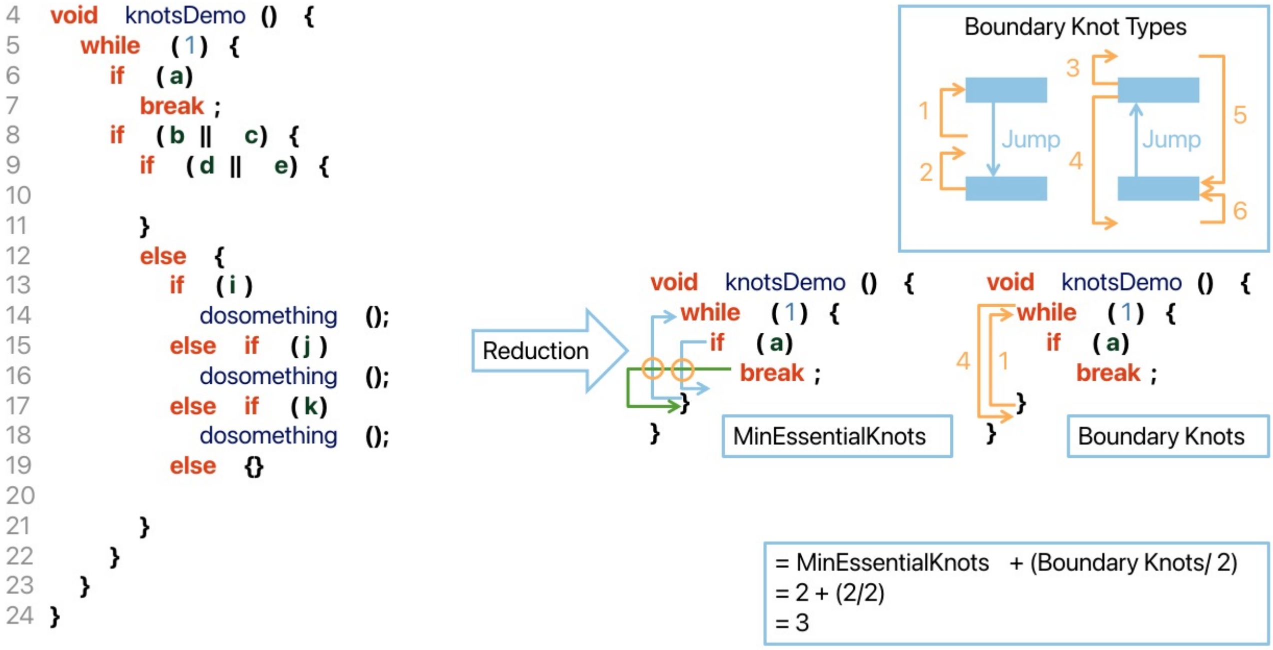
Class: SayHello
= Max(1,3)
= 3

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

Max Essential Knots

API Name: MaxEssentialKnots
Description: Maximum Knots after structured programming constructs have been removed.
Available For:

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



```
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

Max Strict Modified Essential Complexity

API Name: MaxEssentialStrictModified
Description: Maximum strict modified essential complexity of all nested functions or methods.
Available For:

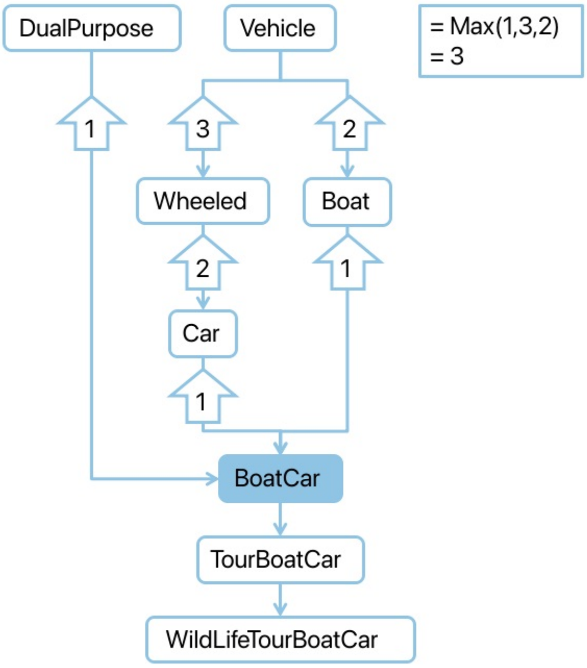
- **Ada:** Project,File,Package

Max Inheritance Tree

API Name: MaxInheritanceTree
Research Name: Chidamber & Kemerer - Depth of Inheritance Tree (DIT)
Description: Maximum depth of class in inheritance tree. [aka DIT (depth of inheritance tree)]

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

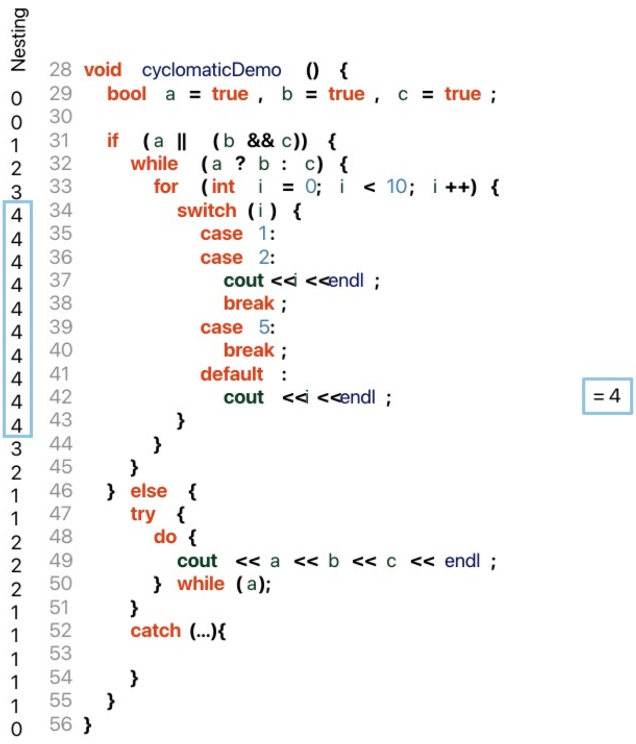
- **C/C++:** Class,Struct,Union
- **C#:** Class,Struct
- **Java:** Class,Interface
- **Pascal:** Class,Interface
- **Python:** Class
- **Web:** PHP Class,PHP Interface



Max Nesting

API Name: MaxNesting
Description: Maximum nesting level of control constructs (if, while, for, switch, etc.) in the function.
Available For:

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

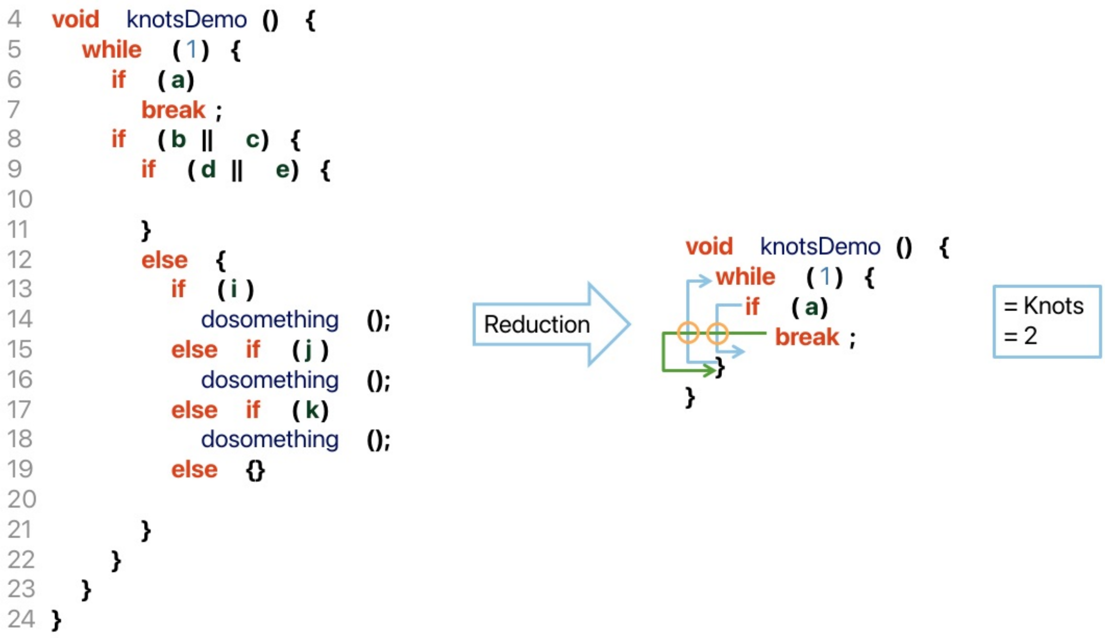


```
Nesting
     void cyclomaticDemo () {
0      bool a = true , b = true , c = true ;
0
1      if (a || (b && c)) {
2        while (a ? b : c) {
3          for (int i = 0; i < 10; i ++) {
4            switch (i) {
4              case 1:
4              case 2:
4                cout <<i <<endl ;
4   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

Min Essential Knots

API Name: MinEssentialKnots
Description: Minimum Knots after structured programming constructs have been removed.
Available For:

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



```
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  {}

         }
      }
   }
}
```

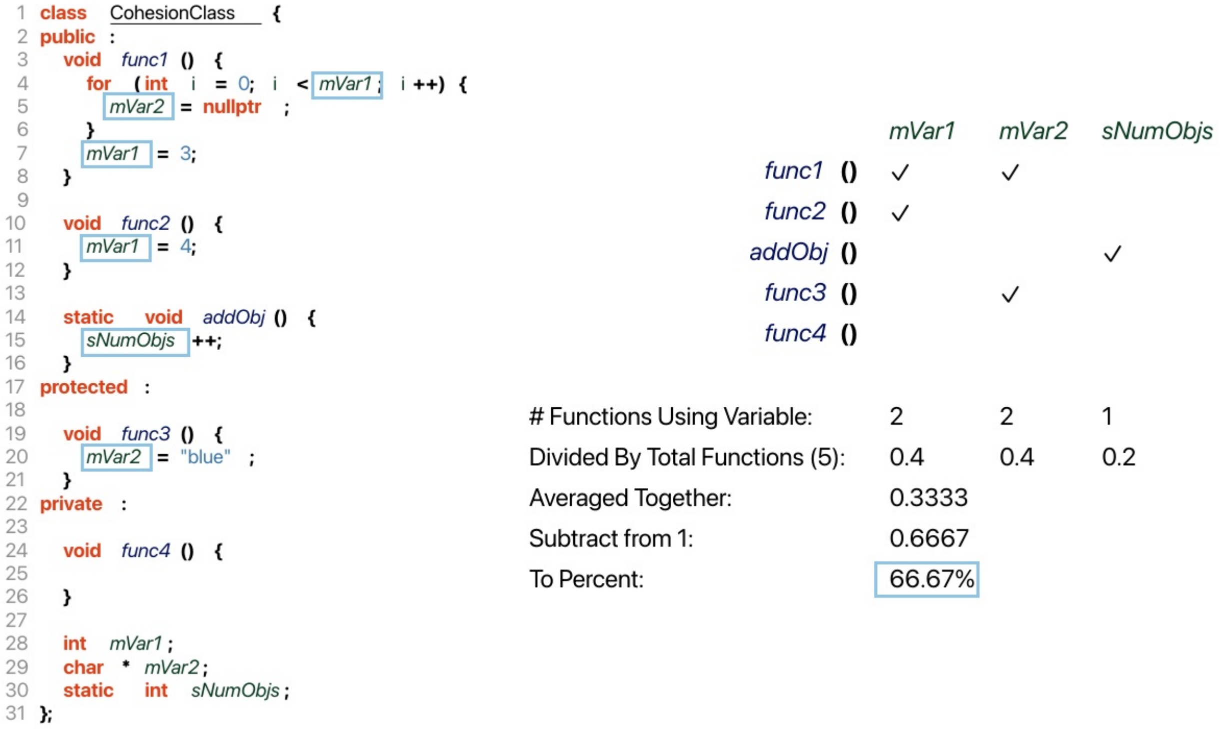
```
void  knotsDemo ()  {
   while  (1)  {
      if  (a)
         break ;
   }
}
```

Percent Lack Of Cohesion

API Name: PercentLackOfCohesion
Research Name: Chidamber & Kemerer - Lack of Cohesion in Methods (LCOM/LOCM), LCOM2
Description: 100% minus the average cohesion for package entities. [aka LCOM, LOCM]

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

- **Ada:** Package
- **Basic:** Class,Struct
- **C/C++:** Class,Struct,Union
- **C#:** Class,Struct
- **Java:** Class,Interface
- **Pascal:** Class,Interface



```
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 ()			

	mVar1	mVar2	sNumObjs
# 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:	66.67%		

Percent Lack Of Cohesion Modified

API Name: PercentLackOfCohesionModified
Description: 100% minus the average cohesion for class data members, modified for accessor methods.

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

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

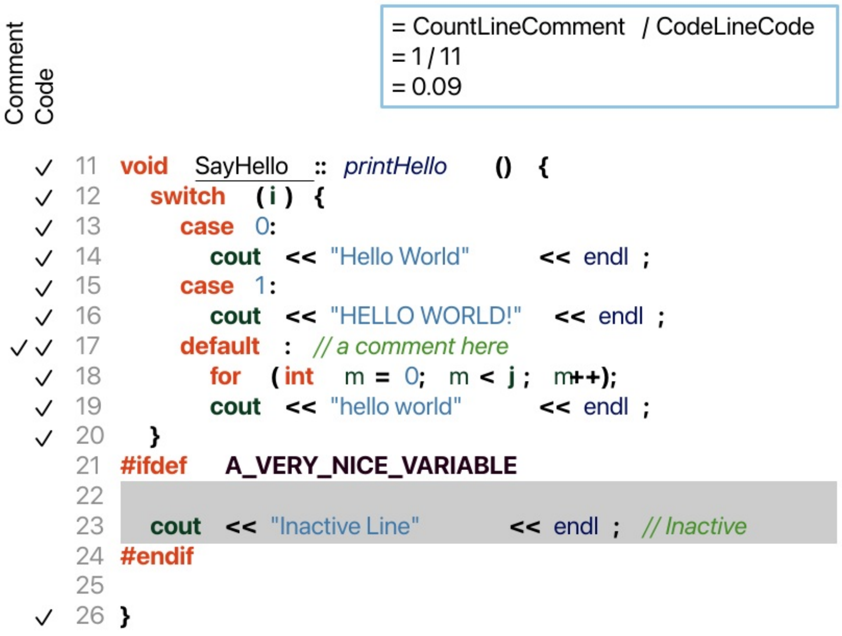
Comment to Code Ratio

API Name: RatioCommentToCode
Description: Ratio of comment lines to code lines.

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

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

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



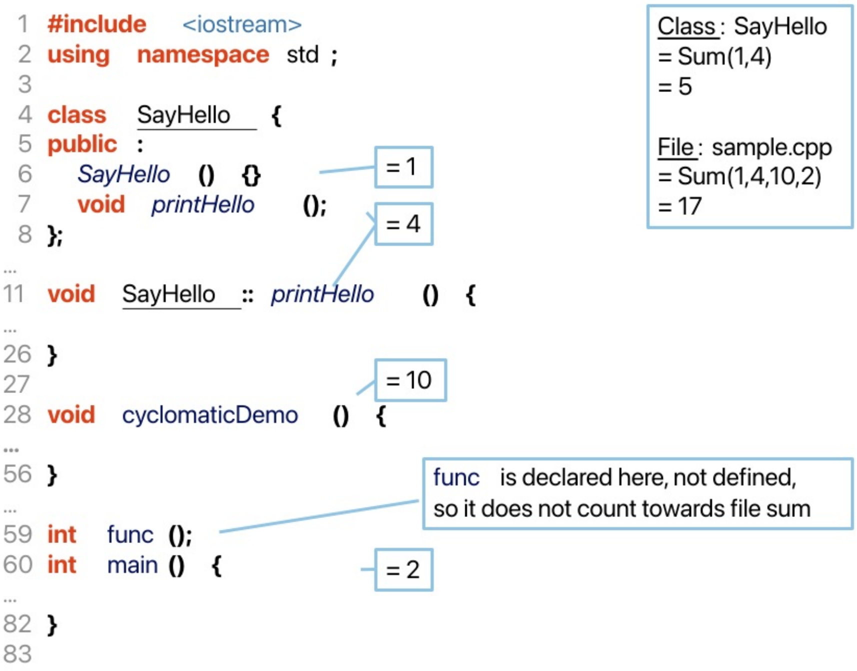
= CountLineComment / CodeLineCode
= 1 / 11
= 0.09

```
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

}
```

Sum Cyclomatic Complexity

API Name: SumCyclomatic
Research Name: Chidamber & Kemerer - Weighted Methods per Class (WMC)
Description: Sum of cyclomatic complexity of all nested functions or methods. [aka WMC (weighted methods per class)]
Available For:

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



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

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

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

```

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

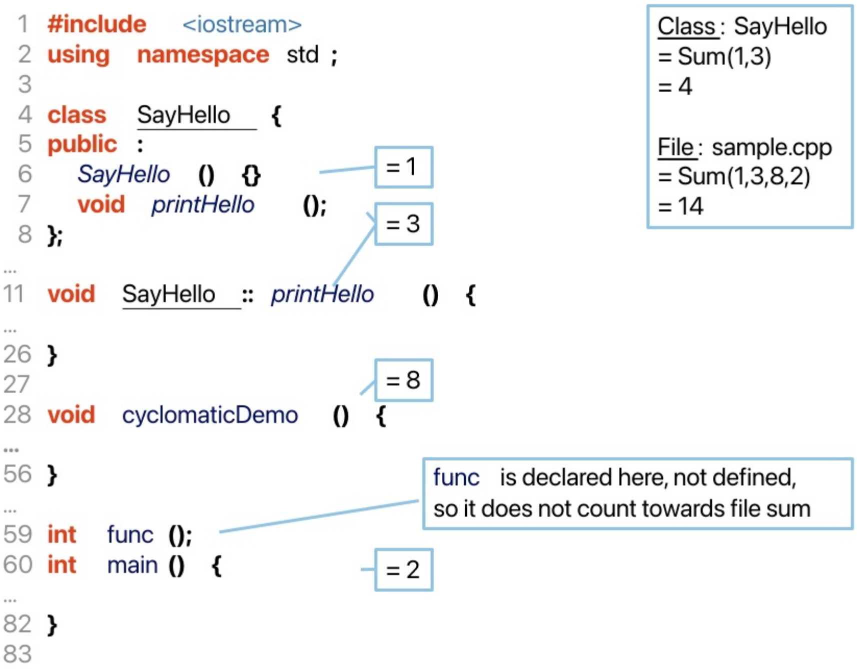
File: sample.cpp
= Sum(1,4,10,2)
= 17

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

Sum Modified Cyclomatic Complexity

API Name: SumCyclomaticModified
Description: Sum of modified cyclomatic complexity of all nested functions or methods.
Available For:

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



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

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

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

= 1

= 3

= 8

= 2

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

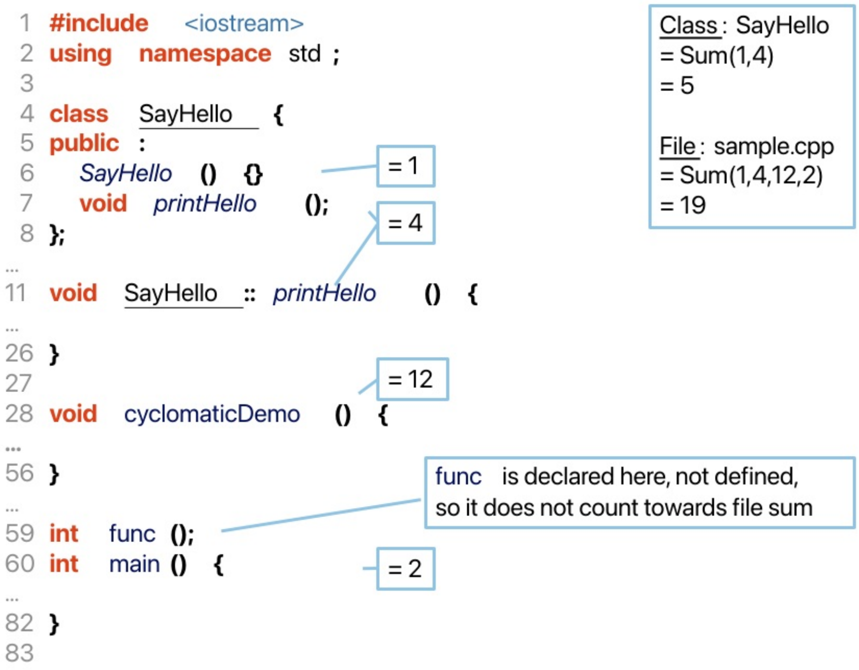
Class: SayHello
= Sum(1,3)
= 4

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

Sum Strict Cyclomatic Complexity

API Name: SumCyclomaticStrict
Description: Sum of strict cyclomatic complexity of all nested functions or methods.
Available For:

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



```
#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 sum

= 2

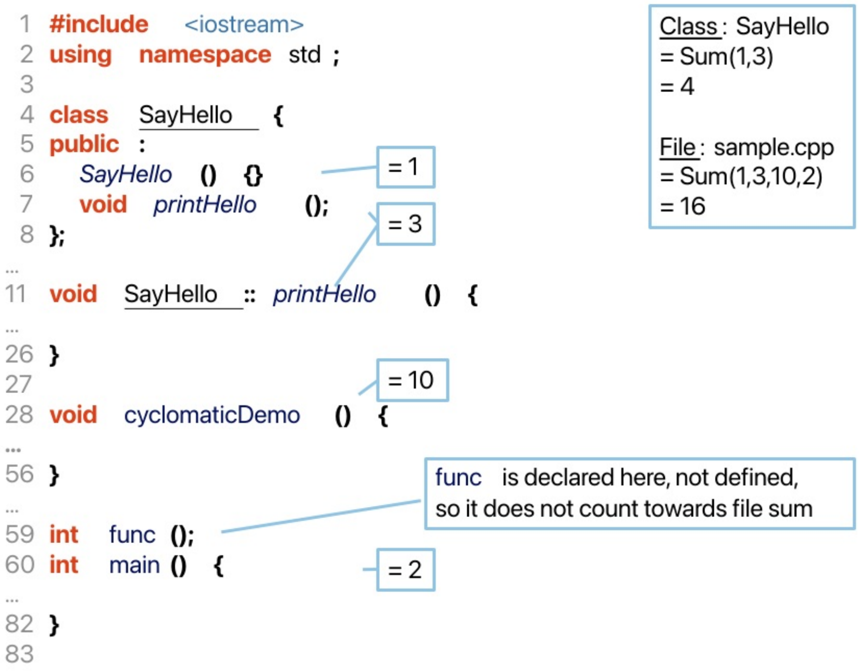
Class: SayHello
= Sum(1,4)
= 5

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

Sum Strict Modified Cyclomatic Complexity

API Name: SumCyclomaticStrictModified
Description: Sum of strict modified cyclomatic complexity of all nested functions or methods.
Available For:

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



```
#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(1,3,10,2)
= 16

= 1

= 3

= 10

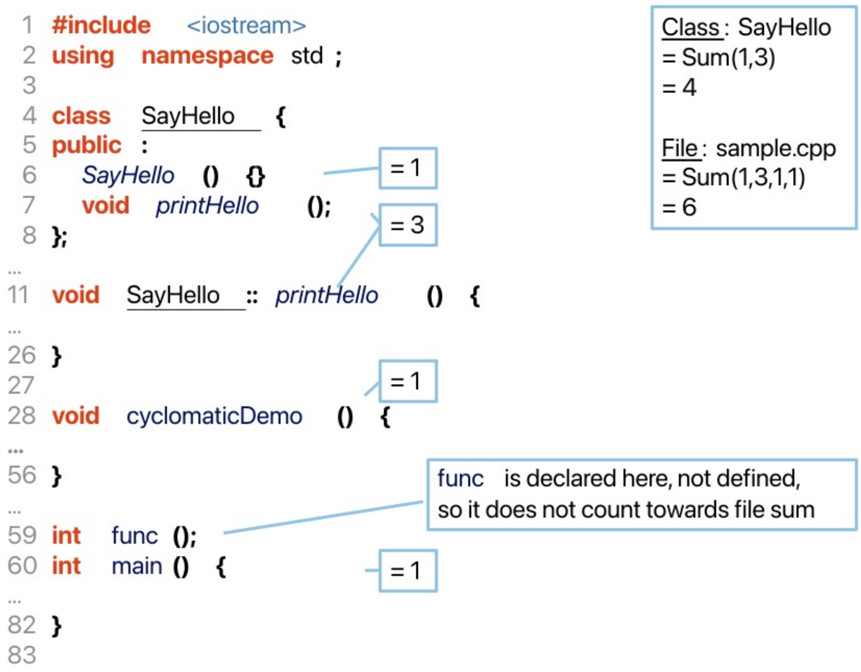
func is declared here, not defined, so it does not count towards file sum

= 2

Sum Essential Complexity

API Name: SumEssential
Description: Sum of essential complexity of all nested functions or methods.
Available For:

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



```
#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

Sum Strict Modified Essential Complexity

API Name: SumEssentialStrictModified
Description: Sum of strict modified essential complexity of all nested functions or methods.
Available For:

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