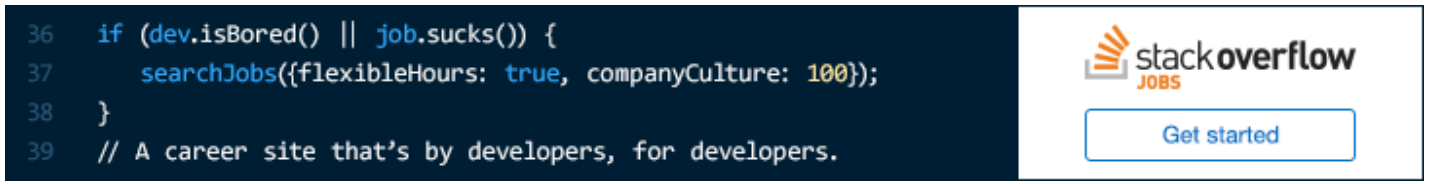
Forward Declaration of Class, Function



When forward declarations of functions work in a source file (.cpp), why would the same doesn't work for classes ?

Thanks.

```
// main.cpp

void forwardDeclaredFunction() ; // This is correct

class One ; // Why this would be wrong

int One:: statVar = 10 ;
```

```
void
One :: anyAccess() {

 std::cout << "\n statVar:\t " << statVar ;
 std::cout << "\n classVar:\t" << classVar ;
}

class One {

 public:
  void anyAccess() ;
  static int statVar ;

 private:
  int  classVar ;

} ;


int main (int argc, char * const argv[]) {

 One *obj = new One ;

        return 0;
}

void forwardDeclaredFunction() {
}
```

c++ visual-c++

edited Dec 2 '10 at 23:50
Zeke
1,252 11 31

asked Dec 2 '10 at 23:45
Mahesh
25.5k 11 64 91

3 `class One;` is the correct syntax for a forward class declaration. What compiler error message are you getting, exactly? – In silico Dec 2 '10 at 23:49

1 error : Invalid use of incomplete type Struct One ; I am running on Mac – Mahesh Dec 2 '10 at 23:53

6 Answers

Forward declaration can work for classes too:

```
class Foo;

class Bar {
public:
    Foo *myFoo; // This has to be a pointer, thanks for catching this!
};

class Foo {
public:
    int value;
};
```

The above code shows a forward declaration of the Foo class, using a variable of type Foo* in another class (Bar), then the actual definition of the Foo class. C++ doesn't care if you leave things unimplemented as long as you implement them before using its code. Defining pointers to objects of a certain type is not "using its code."

Quick, dirty reply but I hope it helps.

Edit: Declaring a non-pointer variable of a class thats unimplemented will NOT compile as the replies stated out. Doing so is exactly what I meant by "using its code." In this case, the Foo constructor would be called whenever the Bar constructor is called, given that it has a member variable of type Foo. Since the compiler doesn't know that you plan on implementing Foo later on, it will throw an error. Sorry for my mistake ;).

edited Dec 3 '10 at 0:09

answered Dec 2 '10 at 23:57

The Maniac

1,981 3 12 30

1 AFAIK, forward declarations don't let you have objects of the class so declared -- only pointers, and possibly references. In your example, the compiler couldn't know `sizeof(Foo)` yet, making it impossible to define `Bar`. – cHao Dec 2 '10 at 23:59

2 This won't compile for the same reason OP's code fails: you are using Foo before its full structure is known (gcc will tell you "field 'myFoo' has incomplete type") – Lars Dec 3 '10 at 0:00

1 Don't forget you can also declare a reference, having only forward-declared it's type, e.g: `const Foo& myFoo`. – Moo-Juice Dec 3 '10 at 16:17

The forward declaration `class One;` allows you to refer to the *class itself* but not to any of its members. You have to put all definitions of class members after the full declaration of the class. (Or inside, of course.)

answered Dec 2 '10 at 23:56

zwol
75.4k 20 131 217

Thanks Zack !!! – Mahesh Dec 3 '10 at 0:06

Place your member declaration of your class before the member implementations.

```
class One {

 public:
  void anyAccess() ;
  static int statVar ;

 private:
  int  classVar ;

} ;

int One:: statVar = 10 ;

void
One :: anyAccess() {

 std::cout << "\n statVar:\t " << statVar ;
 std::cout << "\n classVar:\t" << classVar ;
}
```

answered Dec 2 '10 at 23:56

Daniel A. White
128k 24 250 343

You're getting the error message on `int One:: statVar = 10 ;` NOT on the forward declaration, which is fine.

The compiler needs to know the full definition of the class before you can define static members like that - a forward declaration is insufficient (it needs to be able to confirm that the type is correct from the class definition).

You'll need to move your static attribute definition below the class definition.

answered Dec 2 '10 at 23:56

Mark B
79.4k 4 70 147

The compiler reads stuff from beginning to end, and generates code as it goes. (Some compilers may not do this, but they should behave as if they did.) But before the class is defined, the compiler doesn't know that `One::statVar` or `One::anyAccess` should exist, or whether the function is virtual, static, or what. It needs to know that stuff in order to generate code.

answered Dec 2 '10 at 23:57

cHao
58k 10 90 131

---

when you create 2 class & one function can access data from on class to another class then it is a friend function

forword declaration is use to know which class in next

class abc;

class xyz

```
{
data member;
public:
friend void getdata();
other member function
}
class abc
{
data member
public:
friend void getdata();
}
```

answered Jan 7 '11 at 8:59

user565367