Access Control in C++

Now before studying how to define class and its objects, lets first quickly learn what are access modifiers.

Access modifiers in C++ class defines the access control rules. C++ has 3 new keywords introduced, namely,

1. public
2. private
3. protected

These access modifiers are used to set boundaries for availability of members of class be it data members or member functions

Access modifiers in the program, are followed by a colon. You can use either one, two or all 3 modifiers in the same class to set different boundaries for different class members. They change the boundary for all the declarations that follow them.

Public Access Modifier in C++

Public, means all the class members declared under **public** will be available to everyone. The data members and member functions declared public can be accessed by other classes too. Hence there are chances that they might change them. So the key members must not be declared public.

class PublicAccess

{

// public access modifier

public:

int x; // Data Member Declaration

void display(); // Member Function decaration

}

Private Access Modifier in C++

Private keyword, means that no one can access the class members declared **private**, outside that class. If someone tries to access the private members of a class, they will get a **compile time error**. By default class variables and member functions are private.

class PrivateAccess

{

// private access modifier

private:

int x; // Data Member Declaration

void display(); // Member Function decaration

}

Protected Access Modifier in C++

Protected, is the last access specifier, and it is similar to private, it makes class member inaccessible outside the class. But they can be accessed by any subclass of that class. (If class A is **inherited** by class B, then class B is subclass of class A. We will learn about inheritance later.)

class ProtectedAccess

{

// protected access modifier

protected:

int x; // Data Member Declaration

void display(); // Member Function decaration

}