## Friend Function: -

A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class. Even though the prototypes for friend functions appear in the class definition, friends are not member functions.

A friend can be a function, function template, or member function, or a class or class template, in which case the entire class and all its members are friends.

To declare a function as a friend of a class, precede the function prototype in the class definition with keyword friend as follows −

```
class Box {
double width;
public:

double length;

friend void printWidth( Box box );

void setWidth( double wid );

};
```

To declare all member functions of class ClassTwo as friends of class ClassOne, place a following declaration in the definition of class ClassOne.

## Inheritance in C++

One of the most important concepts in object-oriented programming is that of inheritance. Inheritance allows us to define a class in terms of another class, which makes it easier to create and maintain an application. This also provides an opportunity to reuse the code functionality and fast implementation time.

When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class. This existing class is called the base class, and the new class is referred to as the derived class.

The idea of inheritance implements the is a relationship. For example, mammal IS-A animal, dog IS-A mammal hence dog IS-A animal as well and so on.

## Base and Derived Classes

A class can be derived from more than one classes, which means it can inherit data and functions from multiple base classes. To define a derived class, we use a class derivation list to specify the base class(es). A class derivation list names one or more base classes and has the form −

**class derived-class: access-specifier base-class**

Where access-specifier is one of public, protected, or private, and base-class is the name of a previously defined class. If the access-specifier is not used, then it is private by default.

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

// Base
class class
Shape {

public:

void setWidth(int w) {
width = w;

}

void setHeight(int h) {
height = h;

}


protected:


int width;
int height;

};


// Derived class

class Rectangle: public Shape {
public:

int getArea() {

return (width * height);

}

};


int main(void) {
Rectangle Rect;

Rect.setWidth(5)
;
Rect.setHeight(7)
;

// Print the area of the object.

cout << "Total area: " << Rect.getArea() << endl;

return 0;

}
```

Summary of Access type in Inheritance using different access specifiers.

| Access | public | protected | Private |
|---|---|---|---|
| Same class | yes | yes | Yes |
| Derived classes | yes | yes | No |
| Outside classes | yes | no | No |

- Constructors, destructors and copy constructors of the base class.
- Overloaded operators of the base class.

The friend functions of the base class.