C++ CLASS ACCESS MODIFIERS

Data hiding is one of the important features of Object Oriented Programming which allows preventing the functions of a program to access directly the internal representation of a class type. The access restriction to the class members is specified by the labeled **public**, **private**, and **protected** sections within the class body. The keywords public, private, and protected are called access specifiers.

A class can have multiple public, protected, or private labeled sections. Each section remains in effect until either another section label or the closing right brace of the class body is seen. The default access for members and classes is private.

```
class Base {

   public:

  // public members go here

   protected:

  // protected members go here

   private:

  // private members go here

};
```

The public members:

A **public** member is accessible from anywhere outside the class but within a program. You can set and get the value of public variables without any member function as shown in the following example:

```
#include <iostream>

using namespace std;

class Line
{
   public:
      double length;
      void setLength( double len );
      double getLength( void );
};

// Member functions definitions
double Line::getLength(void)
{
    return length ;
}

void Line::setLength( double len )
{
    length = len;
}

// Main function for the program
int main( )
{
   Line line;

   // set line length
   line.setLength(6.0);
   cout << "Length of line : " << line.getLength() <<endl;

   // set line length without member function
   line.length = 10.0; // OK: because length is public
```

```
   cout << "Length of line : " << line.length <<endl;
   return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
Length of line : 6
Length of line : 10
```

The private members:

A **private** member variable or function cannot be accessed, or even viewed from outside the class. Only the class and friend functions can access private members.

By default all the members of a class would be private, for example in the following class **width** is a private member, which means until you label a member, it will be assumed a private member:

```
class Box
{
   double width;
   public:
      double length;
      void setWidth( double wid );
      double getWidth( void );
};
```

Practically, we define data in private section and related functions in public section so that they can be called from outside of the class as shown in the following program.

```
#include <iostream>

using namespace std;

class Box
{
   public:
      double length;
      void setWidth( double wid );
      double getWidth( void );

   private:
      double width;
};

// Member functions definitions
double Box::getWidth(void)
{
    return width ;
}

void Box::setWidth( double wid )
{
    width = wid;
}

// Main function for the program
int main( )
{
   Box box;

   // set box length without member function
   box.length = 10.0; // OK: because length is public
   cout << "Length of box : " << box.length <<endl;

   // set box width without member function
   // box.width = 10.0; // Error: because width is private
   box.setWidth(10.0);  // Use member function to set it.
   cout << "Width of box : " << box.getWidth() <<endl;
```

```
   return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
Length of box : 10
Width of box : 10
```

The protected members:

A **protected** member variable or function is very similar to a private member but it provided one additional benefit that they can be accessed in child classes which are called derived classes.

You will learn derived classes and inheritance in next chapter. For now you can check following example where I have derived one child class **SmallBox** from a parent class **Box**.

Following example is similar to above example and here **width** member will be accessible by any member function of its derived class SmallBox.

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

class Box
{
   protected:
      double width;
};

class SmallBox:Box // SmallBox is the derived class.
{
   public:
      void setSmallWidth( double wid );
      double getSmallWidth( void );
};

// Member functions of child class
double SmallBox::getSmallWidth(void)
{
    return width ;
}

void SmallBox::setSmallWidth( double wid )
{
    width = wid;
}

// Main function for the program
int main( )
{
   SmallBox box;

   // set box width using member function
   box.setSmallWidth(5.0);
   cout << "Width of box : "<< box.getSmallWidth() << endl;

   return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
Width of box : 5
```