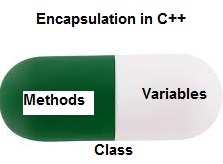
**Encapsulation:**

**Encapsulation**is defined as wrapping up of data and information under a single unit. In Object Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulates them.  
Consider a real-life example of encapsulation, in a company there are different sections like the accounts section, finance section, sales section etc. The finance section handles all the financial transactions and keep records of all the data related to finance. Similarly, the sales section handles all the sales related activities and keep records of all the sales. Now there may arise a situation when for some reason an official from finance section needs all the data about sales in a particular month. In this case, he is not allowed to directly access the data of sales section. He will first have to contact some other officer in the sales section and then request him to give the particular data. This is what encapsulation is. Here the data of sales section and the employees that can manipulate them are wrapped under a single name “sales section”.



Encapsulation also led to data abstraction or hiding. As using encapsulation also hides the data. In the above example the data of any of the section like sales, finance or accounts is hidden from any other section.

In C++ encapsulation can be implemented using Class and access modifiers. Look at the below program:

|  |
| --- |
| // C++ program to explain Encapsulation    #include<iostream>  **using** **namespace** std;  **class** Encapsulation  {  **private**:          // data hidden from outside world  **int** x;    **public**:          // function to set value of          // variable x  **void** set(**int** a)          {              x =a;          }            // function to return value of          // variable x  **int** get()          {  **return** x;          }  };    // main function  **int** main()  {      Encapsulation obj;            obj.set(5);            cout<<obj.get();  **return** 0;  } |

output:

5

In the above program the variable **x** is made private. This variable can be accessed and manipulated only using the functions get() and set() which are present inside the class. Thus we can say that here, the variable x and the functions get() and set() are combined together which is nothing but encapsulation.

**Role of access specifiers in encapsulation**

As we have seen in above example, access specifiers play an important role in implementing encapsulation in C++. The process of implementing encapsulation can be sub-divided into two steps:

1. The data members should be labelled as private using the **private** access specifiers
2. The member function which manipulates the data members should be labelled as public using the **public**access specifier