Data Abstraction in C++

In C++ program if we implement class with private and public members then it is an example of data abstraction.

Data abstraction refers to providing only essential information to the outside world and hiding their background details, i.e., to represent the needed information in program without presenting the details.

#include <iostream>

**using** **namespace** std;

**class** Sum

{

**private**: **int** x, y, z;

**public**:

**void** add()

{

cout<<"Enter two numbers: ";

cin>>x>>y;

z= x+y;

cout<<"Sum of two number is: "<<z<<endl;

}

};

**int** main()

{

Sum sm;

sm.add();

**return** 0;

}

String:

#include <cstring>

string comparison using strcmp() function.

string concatenation using strcat() function.

copy the string using strcpy() function

tring length using strlen() function.

int main()

{

char ch[50]="amit";

char ch1[50]="kumar";

char ch2[50];

int len;

cout<<strcmp(ch,ch1)<<endl;

strcat(ch,ch1);

cout<<ch<<endl;

strcpy(ch2,ch1);

cout<<ch2<<endl;

len=strlen(ch1);

cout<<len;

return 0 ;

}

Namespace:

Namespaces in C++ are used to organize too many classes so that it can be easy to handle the application.

## **Defining a Namespace**

A namespace definition begins with the keyword **namespace** followed by the namespace name as follows −

namespace namespace\_name {

// code declarations

}

To call the namespace-enabled version of either function or variable, prepend (::) the namespace name as follows −

name::code; // code could be variable or function.

## **The using directive**

You can also avoid prepending of namespaces with the **using namespace**directive. This directive tells the compiler that the subsequent code is making use of names in the specified namespace