Tutorial Link https://codequotient.com/tutorials/Friend Functions & Classes/5b38c3bdc6a1d0259e728e6b

TUTORIAL

Friend Functions & Classes

Chapter

1. Friend Functions & Classes

Topics

1.1 Friend Functions

1.4 Friend Classes

Friend Functions

In C++, a non-member function can access to the private members of a class. This is possible by declaring a non-member function friend to the class whose private members are to be accessed. The syntax of declaration of function is as under:

```
class < class name>
{
        ..........
        public:
        ..........
        friend < return type> <function name>(<argument type>);
};
```

The function declaration should be preceded by the keyword friend. The function is defined as a normal function anywhere in the program outside the class and its definition does not use friend keyword.

The friend functions have certain properties:

1. The friend function is not in the scope of the class to which it is declared; therefore it is called without using object name.
2. It cannot access the members of a class directly. It has to use object and dot operator to access the private and public members of a class. For example obj.member, where obj is the object of a class and member is member function or data member.
3. Generally friend functions have the objects as arguments.
4. It can be declared anywhere in the class without affecting its meaning and scope.
5. Friend function of a class can be a member function of another class.

Let us take an example of 'Account' class to access its private data like Acc_No , Name, Amount to calculate the TDS by using a non-member function TDS_Cal.

C++

```
#include<iostream>
#include<cstdio>
#include<cstring>
// Include headers as needed

using namespace std;

class Account
{
  private:
  char name[18];
  int acc_no;
  float amount;
  public:
  void getData( )
  {
    strcpy(name, "Amit");
    acc_no = 1234567890;
    amount = 34000.00;
```

```
  }
  friend void TDS_Cal(Account);
};

void TDS_Cal (Account ac)
{
  float tds;
  tds = ac.amount/10;
  cout <<"\n Tds of account no." << ac.acc_no << " is Rs. " << tds;
}

int main ( )
{
  Account Acc;
  Acc.getData();
  TDS_Cal(Acc);
  return 0;
}
```

OUTPUT:

```
Tds of account no.1234567890 is Rs. 3400
```

In this program, the class account has three member variables and one member function. Inside the class a friend function TDS_Cal is declared and it has authority to access the private members of the class. The function getData reads the data by keyboard and friend function TDS_Cal calculates and displays the TDS.

Friend Classes

More than one function can be declared as friend functions or an entire class can be declared as friend class. When all the functions of a class need to access another class then that entire class can be

declared as friend class. By default friendship is not mutual i.e., if class A is declared as friend class of class B, this does not mean that class B has privileges to access private members of class A. There is an example given below in which values of data members of both classes are shown by the function of a class by making it friend to another class.

C++

```
#include<iostream>
#include<cstdio>
#include<cmath>
// Include headers as needed

using namespace std;

class Y;
class X
{
  int x;
  public:
  void read_value1( )
  {
    x=25;
  }
  void display(Y);
};

class Y
{
  int y;
  public :
  void read_value2( )
  {
    y = 45;
  }
  friend void X :: display(Y);
};

```

```
31 void X :: display(Y y1)
32 {
33   cout << "x = "<< x << endl;
34   cout << "y = "<< y1.y << endl;
35 }
36
37 int main( )
38 {
39   X x1;
40   Y y1;
41   x1.read_value1( );
42   y1.read_value2( );
43   x1.display(y1);
44   return 0;
45 }
46
```

Output:

```
x=25
y=45
```

In the above program, class X is the friend class of class Y. It means the member function of class X can access the data members of class Y. Therefore the display function of class X shows the values of data members of both classes.