

LECTURE-18

FRIENDLY FUNCTIONS:-

We know private members can not be accessed from outside the class. That is a non-member function can't have an access to the private data of a class. However there could be a case where two classes manager and scientist, have been defined we should like to use a function incometax to operate on the objects of both these classes.

In such situations, c++ allows the common function to be made friendly with both the classes , there by following the function to have access to the private data of these classes .Such a function need not be a member of any of these classes.

To make an outside function "friendly" to a class, we have to simply declare this function as a friend of the classes as shown below :

```
class ABC
{
-----
-----
public:
-----
-----
    friend void xyz(void);
};
```

The function declaration should be preceded by the keyword friend , The function is defined else where in the program like a normal C ++ function . The function definition does not use their the keyword friend or the scope operator :: . The functions that are declared with the keyword friend are known as friend functions. A function can be declared as a friend in any no of classes. A friend function, as though not a member function , has full access rights to the private members of the class.

A friend function processes certain special characteristics:

- a. It is not in the scope of the class to which it has been declared as friend.
- b. Since it is not in the scope of the class, it cannot be called using the object of that class. It can be invoked like a member function without the help of any object.
- c. Unlike member functions.

Example:

```
#include<iostream.h>
class sample
```

```

        { int a;
        int b;
        public:
            void setvalue( ) { a=25;b=40;}
            friend float mean( sample s);
        }
float    mean (sample s)
        { return (float(s.a+s.b)/2.0);
        }
int main ( )
    {
        sample x;
        x
        setvalue(
        );
        cout<<"mean    value="<<mean(x)<<endl;
        return(0);
    }

```

output:
mean value : 32.5

A function friendly to two classes

```

#include<iostrea
    m.h>
class abc;
class xyz
{ int x;
public:
    void setvalue(int x) { x-= I; }
    friend void max (xyz,abc);
};
clas
s
abc
{ int a;
public:
    void setvalue( int i) {a=i; }
    friend void max(xyz,abc);
};

void max( xyz m, abc n)

```

```

{ if(m . x >= n.a)
    cout<<m.x;
  else  cout<<
        n.a;
}

```

```

int main( )
{ abc j; j .
  setvalue(
  10); xyz s;
  s.setvalue(20);
  max( s , j );
  return(0);
}

```

SWAPPING PRIVATE DATA OF CLASSES:

```

#include<iostream.h>
class class-2; class class-1
{
    int value 1;
public:
    void indata( int a) { value=a; } void
display(void) { cout<<value<<endl; } friend
void exchange ( class-1 &, class-2 &); };

class class-2
{ int value2;
public:
    void indata( int a) { value2=a; } void
display(void) { cout<<value2<<endl; }
friend void exchange(class-1 & , class-2
&);
}; void exchange ( class-1
&x, class-2 &y)
{ int temp=x. value 1;
  x. value I=y.valuo2;
  y.value2=temp;
}

int main( )
{ class-1
c1; class-2
c2;
c1.indata(1

```

```

00);
c2.indata(
200);
cout<<"values before exchange:"<<endl;
c1.display( );
c2.display( );
exchange (c1,c2);
cout<<"values after exchange :"<< endl;
c1.
display (
); c2.
display (
);
return(0);
}

```

output:

```

values before exchange
    100
    200
values      after
exchange
    200
    100

```

PROGRAM FOR ILLUSTRATING THE USE OF FRIEND FUNCTION:

```

#include<
iostream.h> class
account1; class
account2
{ private: int
balance;
public: account2( ) {
balance=567; } void
showacc2( )
{
cout<<"balanceinaccount2 is:"<<balance<<endl;
friend int transfer (account2 &acc2, account1 &acc1,int amount);
};
class
account1

```

```

{ private: int
balance;
public:
    account1 ( ) { balance=345; }

    void showacc1 ( )
    {
        cout<<"balance in account1 :"<<balance<<endl;
    }
friend int transfer (account2 &acc2, account1 &acc1 ,int
amount); };

int transfer ( account2 &acc2, account1 & acc1, int amount)
{ if(amount <=acc1 . bvalue)
    { acc2. balance + =
    amount; acc1
    .balance - = amount;
    }
else return(0); } int main()
{ account1 aa;
account2 bb;

```

```

    cout << "balance in the accounts before
transfer:" ; aa . showacc1 ( ); bb . showacc2 ( );
    cout << "amt transferred from account1 to account2 is:";
    cout<<transfer ( bb,aa,100)<<endl;
    cout<< " balance in the accounts after the
transfer:"; aa . showacc 1 ( ); bb. showacc 2 ( );
    return(0);

```

} output:

```

balance in the accounts before
transfer balance in
account 1 is 345 balance
in account2 is 567
and transferred from account! to account2 is
100 balance in account 1 is 245
balance in account2 is 667

```

