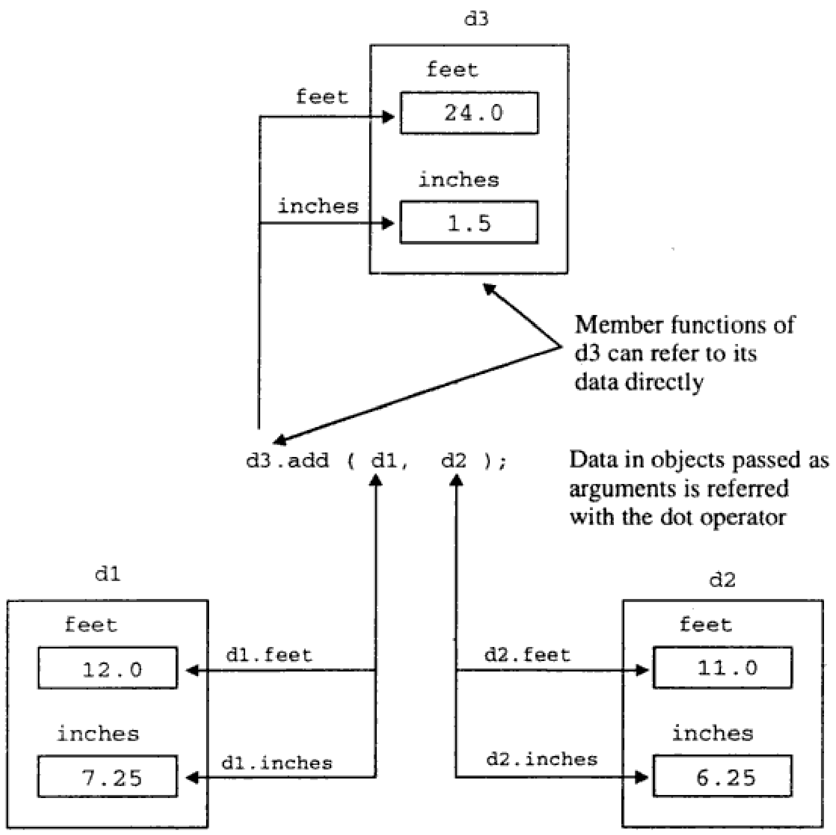
d3
feet
24.0
inches
1.5
feet
inches
Member functions of d3 can refer to its data directly
d3.add (d1, d2);
Data in objects passed as arguments is referred with the dot operator
d1
feet
12.0
inches
7.25
d1.feet
d1.inches
d2
feet
11.0
inches
6.25
d2.feet
d2.inches

Figure 10.16: Objects of the distance class as parameters

```
// account.cpp: passing objects as parameters to functions
#include<iostream.h>
class AccClass
{
   private:                 // class data members
      int accno;
      float balance;
   public:                  // class function members
      void getdata()
      {
         cout << "Enter the account number for acc1 object: ";
         cin >> accno;
         cout << "Enter the balance: ";
         cin >> balance;
      }
      void setdata( int accIn )
      {
         accno = accIn;
         balance = 0;
      }
      void setdata( int accIn, float balanceIn )
      {
         accno = accIn;
         balance = balanceIn;
      }
      void display()
      {
         cout << "Account number is: " << accno << endl;
         cout << "Balance is: " << balance << endl;
      }
      void MoneyTransfer( AccClass & acc, float amount );
};
// acc1.MoneyTransfer( acc2, 100 ),transfers 100 rupees from acc1 to acc2
void AccClass::MoneyTransfer( AccClass & acc, float amount )
{
   balance = balance - amount;   // deduct money from source
   acc.balance = acc.balance + amount; // add money to destination
}
void main()
{
   int trans_money;
   AccClass acc1, acc2, acc3;
   acc1.getdata();
   acc2.setdata( 10 );
   acc3.setdata( 20, 750.5 );
   cout << "Account Information..." << endl;
   acc1.display();
   acc2.display();
   acc3.display();
   cout << "How much money is to be transferred from acc3 to acc1: ";
   cin >> trans_money;
   acc3.MoneyTransfer(acc1,trans_money); //transfers money from acc3 to acc1
   cout << "Updated Information about accounts..." << endl;
   acc1.display();
   acc2.display();
   acc3.display();
}
```

Run

```
Enter the account number for acc1 object: 1
Enter the balance: 100
Account Information...
Account number is: 1
Balance is: 100
Account number is: 10
Balance is: 0
```

```
Account number is: 20
Balance is: 750.5
How much money is to be transferred from acc3 to acc1: 200
Updated Information about accounts...
Account number is: 1
Balance is: 300
Account number is: 10
Balance is: 0
Account number is: 20
Balance is: 550.5
```

In `main()`, the statement

```
acc3.MoneyTransfer( acc1, trans_money );
```

transfers the object `acc1` by reference to the member function `MoneyTransfer()`. It is to be noted that when the `MoneyTransfer()` is invoked with `acc1` as the object parameter, the data members of `acc3` are accessed without the use of the class member access operator, while the data members of `acc1` are accessed by using their names in association with the name of the object to which they belong. An object can also be passed to a non-member function of the class and that can have access to the public members only through the objects passed as arguments to it.

Passing Objects by Pointer

The members of objects passed by pointer are accessed by using the -> operator, and they have similar effect as those passed by value. The above program requires the following changes if parameters are to be passed by pointer:

1. The prototype of the member function `MoneyTransfer()` has to be changed to:

```
void MoneyTransfer( AccClass * acc, float amount );
```

2. The definition of the member function `MoneyTransfer()` has to be changed to:

```
void AccClass::MoneyTransfer( AccClass & acc, float amount )
{
   balance = balance - amount;   // deduct money from source
   acc->balance = acc->balance + amount; // add money to destination
}
```

3. The statement invoking the member function `MoneyTransfer()` has to be changed to:

```
acc3.MoneyTransfer( &acc1, trans_money );
```

10.13 Returning Objects from Functions

Similar to sending objects as parameters to functions, it is also possible to return objects from functions. The syntax used is similar to that of returning variables from functions. The return type of the function is declared as the return object type. It is illustrated in the program `complex.cpp`.

```
// complex.cpp: Addition of Complex Numbers, class complex as data type
#include <iostream.h>
#include <math.h>
class complex
{
   private:
      float real;    // real part of complex number
      float imag;    // imaginary part of complex number
```