```
                cout << "A";
        }
};
class B: public A
{
public:
        B()
        {
                cout << "B";
        }
};
class C: public B
{
public:
        C()
        {
                cout << "C";
        }
};
class D
{
public:
        D()
        {
                cout << "D";
        }
};
class E: public C, public D
{
public:
        E()
        {
                cout << "D";
        }
};
class F: B, virtual E
{
public:
        F()
```

```
        {
                cout << "F";
        }
};
void main()
{
        F f;
}
```

8.3 Identify the error in the following program.

```
#include <iostream.h>
class A
{
        int i;
};

class AB: virtual A
{
        int j;
};
class AC: A, ABAC
{
        int k;
};
class ABAC: AB, AC
{
        int l;
};
void main()
{
        ABAC abac;
        cout << "sizeof ABAC:" << sizeof(abac);
}
```

8.4 Find errors in the following program. State reasons.

```
// Program test
#include <iostream.h>

class X
```

```
{
        private:
           int x1;
        protected:
           int x2;
        public:
           int x3;
};

class Y: public X
{
        public:
           void f()
           {
                 int y1,y2,y3;
                 y1 = x1;
                 y2 = x2;
                 y3 = x3;
           }
};

class Z: X
{
        public:
           void f()
           {
                 int z1,z2,z3;
                 z1 = x1;
                 z2 = x2;
                 z3 = x3;
           }
};
main()
{
              int m,n,p;
              Y y;
              m = y.x1;
              n = y.x2;
              p = y.x3;
              Z z;
              m = z.x1;
              n = z.x2;
              p = z.x3;
}
```

8.5 Debug the following program.

```
// Test program
#include <iostream.h>

class B1
{
        int b1;
  public:
        void display();
        {
                cout << b1 <<"\n";
        }
};

class B2
{
        int b2;
  public:
        void display();
        {
                cout << b2 <<"\n";
        }
};
class D: public B1, public B2
{
        // nothing here
};
main()
{
        D d;
        d.display()
        d.B1::display();
        d.B2::display();
}
```

Programming Exercises

8.1 *Assume that a bank maintains two kinds of accounts for customers, one called as savings account and the other as current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.*

Create a class ***account*** *that stores customer name, account number and type of account. From this derive the classes* ***cur_acct*** *and* ***sav_acct*** *to make them more specific to their requirements. Include necessary member functions in order to achieve the following tasks:*

(a) *Accept deposit from a customer and update the balance.*
(b) *Display the balance.*
(c) *Compute and deposit interest.*
(d) *Permit withdrawal and update the balance.*
(e) *Check for the minimum balance, impose penalty, necessary, and update the balance.*

Do not use any constructors. Use member functions to initialize the class members.

8.2 *Modify the program of Exercise 8.1 to include constructors for all the three classes.*

8.3 *An educational institution wishes to maintain a database of its employees. The database is divided into a number of classes whose hierarchical relationships are shown in Fig. 8.14. The figure also shows the minimum information required for each class. Specify all the classes and define functions to create the database and retrieve individual information as and when required.*

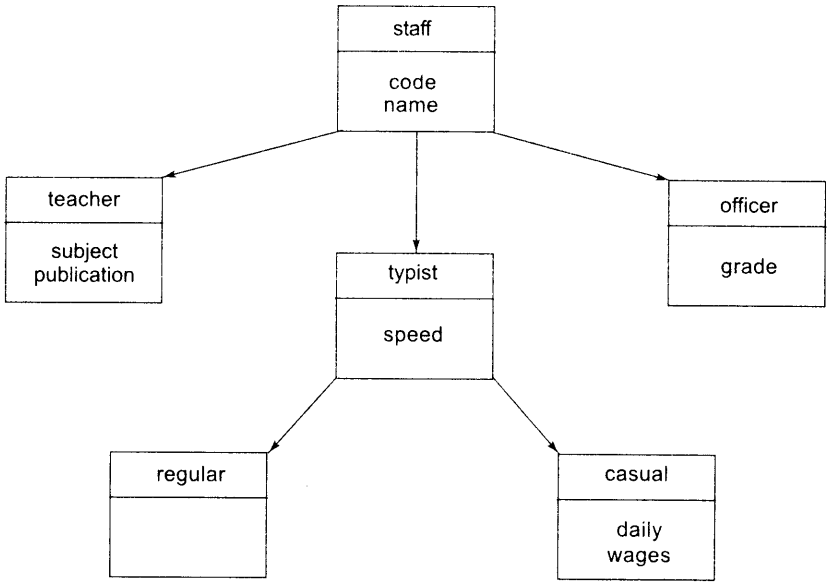

Fig. 8.14 ⇔ *Class relationships (for Exercise 8.19)*

8.4 *The database created in Exercise 8.3 does not include educational information of the staff. It has been decided to add this information to teachers and officers (and not for typists) which will help the management in decision making with regard to training, promotion, etc. Add another data class called* ***education*** *that holds*

two pieces of educational information, namely, highest qualification in general education and highest professional qualification. This class should be inherited by the classes ***teacher*** *and* ***officer****. Modify the program of Exercise 8.19 to incorporate these additions.*

8.5 *Consider a class network of Fig. 8.15. The class* ***master*** *derives information from both account and admin classes which in turn derive information from the class* ***person****. Define all the four classes and write a program to create, update and display the information contained in* ***master*** *objects.*

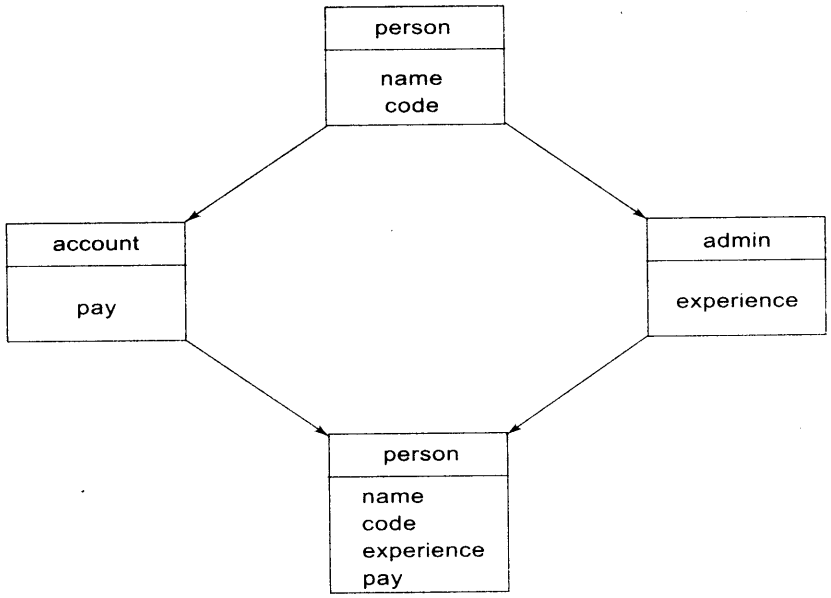

Fig. 8.15 ⇔ *Multipath inheritance (for Exercise 8.21)*

8.6 *In Exercise 8.3, the classes* ***teacher***, ***officer***, *and* ***typist*** *are derived from the class* ***staff****. As we know, we can use container classes in place of inheritance in some situations. Redesign the program of Exercise 8.3 such that the classes* ***teacher***, ***officer***, *and* ***typist*** *contain the objects of* ***staff****.*

8.7 *We have learned that OOP is well suited for designing simulation programs. Using the techniques and tricks learned so far, design a program that would simulate a simple real-world system familiar to you.*