```
.....
.....
        outfile.close();              // Disconnect salary from outfile
        ifstream infile("salary");    // and connect to infile
.....
.....
        infile.close();               // Disconnect salary from infile
```

Although we have used a single program, we created two file stream objects, **outfile** (to put data into the file) and **infile** (to get data from the file). Note that the use of a statement like

```
outfile.close();
```

disconnects the file salary from the output stream **outfile**. Remember, the object **outfile** still exists and the **salary** file may again be connected to **outfile** later or to any other stream. In this example, we are connecting the **salary** file to **infile** stream to read data.

Program 11.1 uses a single file for both writing and reading the data. First, it takes data from the keyboard and writes it to the file. After the writing is completed, the file is closed. The program again opens the same file, reads the information already written to it and displays the same on the screen.

Program 11.1 Working with Single File

```
// Creating files with constructor function

#include <iostream.h>
#include <fstream.h>

int main()
{
        ofstream outf("ITEM");        // connect ITEM file to outf

        cout << "Enter item name:";
        char name[30];
        cin >> name;                  // get name from key board and
        outf << name << "\n";         // write to file ITEM

        cout << "Enter item cost:";
        float cost;
        cin >> cost;                  // get cost from key board and
        outf << cost << "\n";         // write to file ITEM
        outf.close();                 // Disconnect ITEM file from outf

        ifstream inf("ITEM");         // connect ITEM file to inf

        inf >> name;                  // read name from file ITEM
```

(Contd.)

```
        fin >> cost;                    // read cost from file ITEM
        cout << "\n";
        cout << "Item name:" << name << "\n";
        cout << "Item cost:" << cost << "\n";

        fin.close();                    // Disconnect ITEM from inf

        return 0;
}
```

The output of Program 11.1 would be:

```
Enter item name:CD-ROM
Enter item cost:250

Item name:CD-ROM
Item cost:250
```

Opening Files Using open()

As stated earlier, the function **open()** can be used to open multiple files that use the same stream object. For example, we may want to process a set of files sequentially. In such cases, we may create a single stream object and use it to open each file in turn. This is done as follows:

```
file-stream-class stream-object;
stream-object.open ("filename");
```

Example:

```
ofstream outfile;                   // Create stream (for output)
outfile.open("DATA1");              // Connect stream to DATA1
.....
.....
outfile.close();                    // Disconnect stream from DATA1
outfile.open("DATA2");              // Connect stream to DATA2
.....
.....
outfile.close();                    // Disconnect stream from DATA2
.....
.....
```

The previous program segment opens two files in sequence for writing the data. Note that the first file is closed before opening the second one. This is necessary because a stream can be connected to only one file at a time. See Program 11.2 and Fig. 11.6.

Program 11.2 Working with Multiple Files

```
// Creating files with open() function

#include <iostream.h>
#include <fstream.h>

int main()
{
    ofstream fout;
    fout.open("country");           // create output stream
                                    // connect "country" to it

    fout << "United States of America\n";
    fout << "United Kingdom\n";
    fout << "South Korea\n";

    fout.close();                   // disconnect "country" and
    fout.open("capital");           // connect "capital"

    fout << "Washington\n";
    fout << "London\n";
    fout << "Seoul\n";

    fout.close();                   // disconnect "capital"

    // Reading the files
    const int N = 80;               // size of line
    char line[N];

    ifstream fin;                   // create input stream
    fin.open("country");            // connect "country" to it

    cout <<"contents of country file\n";

    while(fin)                      // check end-of-file
    {
        fin.getline(line, N);       // read a line
        cout << line ;              // display it
    }

    fin.close();                    // disconnect "country" and
    fin.open("capital");            // connect "capital"

    cout << "\nContents of capital file \n";

    while(fin)
    {
        fin.getline(line, N);
        cout << line ;
```

(Contd.)

```
    }
    fin.close();
    return 0;
}
```

The output of Program 11.2 would be:

```
Contents of country file
United States of America
United Kingdom
South Korea

Contents of capital file
Washington
London
Seoul
```

LIMITATION

There is a limitation of opening a number of files simultaneously. However, this is not limited in C++; it is dependent on operating systems (like in Windows: 2048 files; in Windows XP: 65535 files).

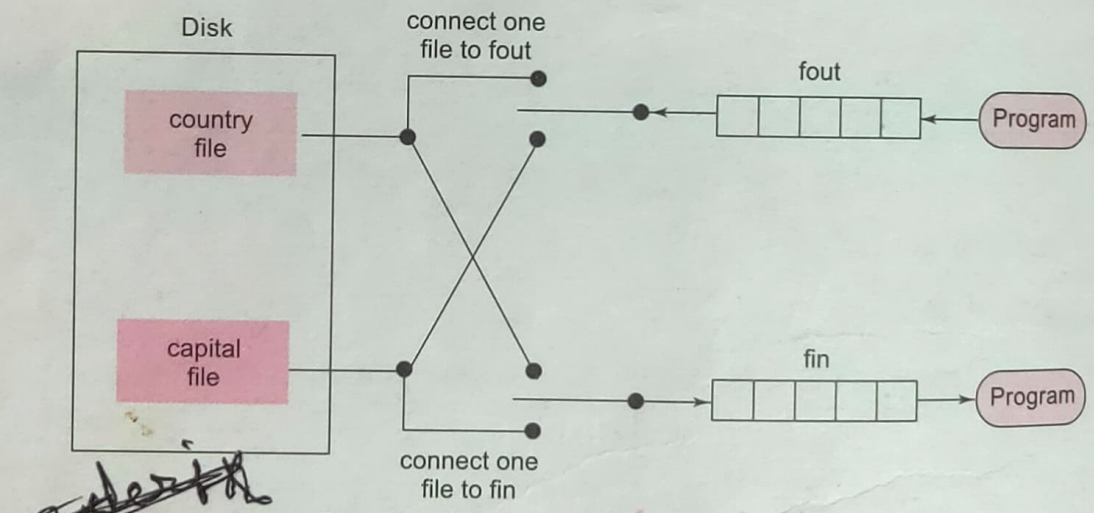

Fig. 11.6 *Streams working on multiple files*

At times we may require to use two or more files simultaneously. For example, we may require to merge two sorted files into a third sorted file. This means, both the sorted files have to be kept open for reading and the third one kept open for writing. In such cases, we need to create two separate input streams for handling the two input files and one output stream for handling the output file. See Program 11.3.

Program 11.3 Reading from Two Files Simultaneously

```
// Reads the files created in Program 11.2

#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>                    // for exit() function
```

(Contd.)

```
int main()
{
        const int SIZE = 80;
        char line[SIZE];

        ifstream fin1, fin2;
        fin1.open("country");
        fin2.open("capital");

        for(int i=1; i<=10; i++)
        {
           if(fin1.eof() != 0)
           {
             cout << "Exit from country \n";
             exit(1);
           }

           fin1.getline(line, SIZE);
           cout << "Capital of "<< line;

           if(fin2.eof() != 0)
           {
             cout << "Exit from capital\n";
             exit(1);
           }

           fin2.getline(line,SIZE);
           cout << line << "\n";
        }
return 0;
}
```

The output of Program 11.3 would be:

```
Capital of United States of America
Washington
Capital of United Kingdom
London
Capital of South Korea
Seoul
```

11.4 DETECTING END-OF-FILE

Detection of the end-of-file condition is necessary for preventing any further attempt to read data from the file. This was illustrated in Program 11.2 by using the statement

```
while(fin)
```

An **ifstream** object, such as **fin**, returns a value of 0 if any error occurs in the file operation including the end-of-file condition. Thus, the **while** loop terminates when **fin** returns a value of zero on reaching the end-of-file condition. Remember, this loop may terminate due to other failures as well. (We will discuss other error conditions later.)