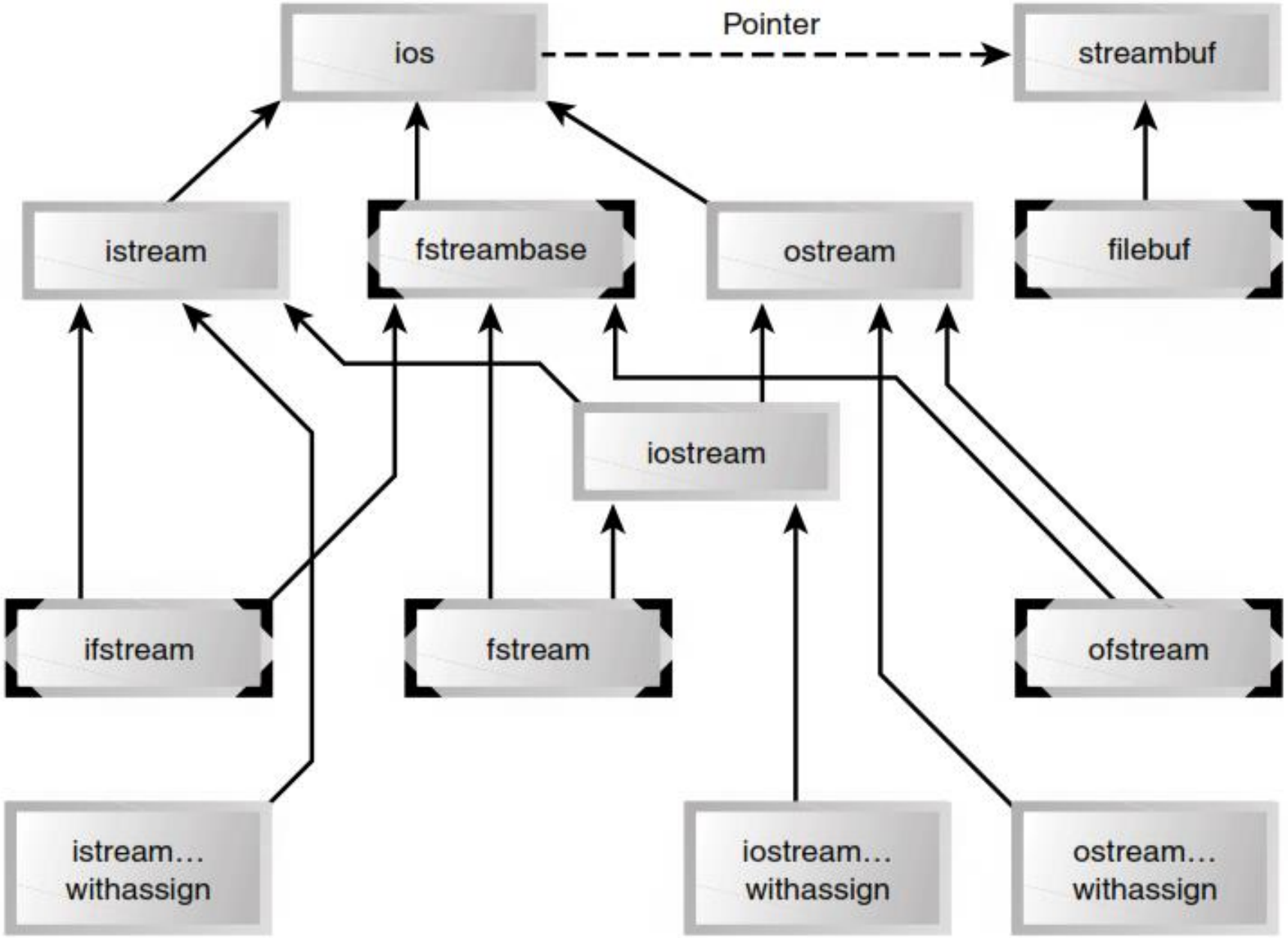
Files and Streams

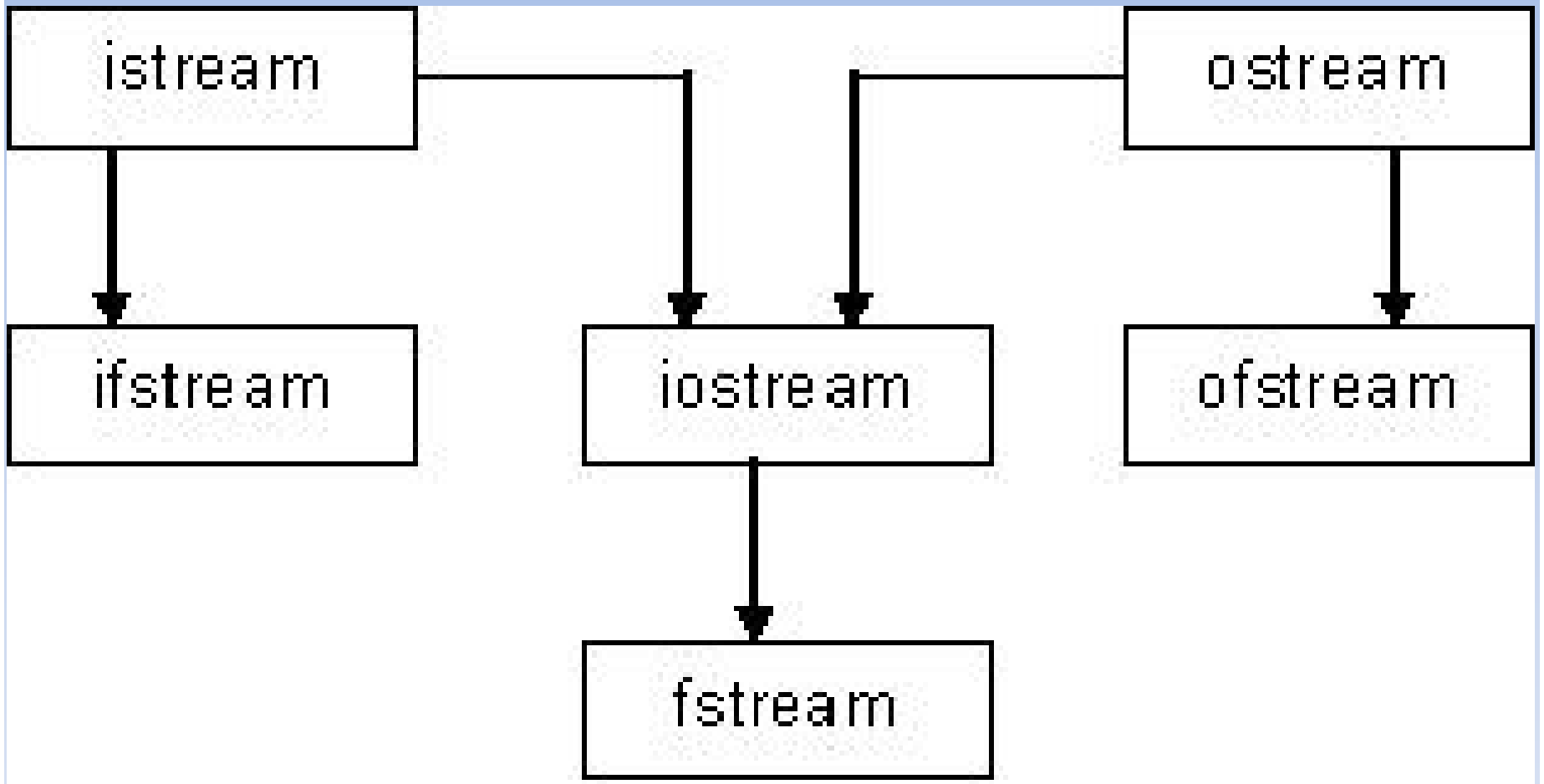
Streams

- Stream
  - A transfer of information in the form of a sequence of bytes

- Input stream
  - Flow into program
    - Can come from keyboard
    - Can come from file
- Output stream
  - Flow out of program
    - Can go to screen
    - Can go to file


ios
Pointer
streambuf
istream
fstreambase
ostream
filebuf
iostream
ifstream
fstream
ofstream
istream…
withassign
iostream…
withassign
ostream…
withassign

Stream Classes



Istream and ostream

- The istream and ostream classes are derived from ios and are dedicated to input and output, respectively.
- The istream class contains such functions as get(), getline(), read(), and the overloaded extraction (>>) operators,
- while ostream contains put() and write(), and the overloaded insertion (<<) operators.

The extraction operator >> is a member of the istream class, and the insertion operator << is a member of the ostream class.

C++ file I/O

- ifstream in; // *Provides input operations on files*
- ofstream out; // *Provides output operations on files*
- fstream io; // *supports both input and output operations on files*

Example:

```
ifstream   mystream("myfile");   // open file for input

ofstream   mystream("myfile");   // open file for output
```

Example-1: File Output

```
1  #include<iostream>
2  #include <fstream> //Required for file I/O
3  using namespace std;
4  int main()
5  {
6      ofstream myfile ("message.txt");
7      if (!myfile)
8      {
9          //check if the file is opened or not
10         cout<<"\n Cannot open this file";
11         return 1;
12     }
13     myfile<<"When an apple fell, Newton was disturbed \n";
14     myfile<<"but, when he found that all apples fell, \n";
15     myfile<<"it was gravitation that attracts them down,\n";
16     myfile<<"he was satisfied \n";
17     myfile.close();                //close the file
18 }
```

Example-2: File Input

```
4   int main()
5   {
6       ifstream myfile ("message.txt");
7       char str[75];
8       if ( !myfile )
9       {
10          //check if the file is opened or not
11          cout<<"\n Cannot open this file";
12          exit(0);
13      }
14      while( 1 )
15      {
16          myfile >> str;
17          if(myfile.eof())
18              break;
19          cout << str;
20      }
21      myfile.close();  //close the file
22  }
```

Question-1

```
int main()
{
    ofstream op_file("test.txt");

    op_file<< "ONE TWO THREE \n";
    op_file<< "FOUR FIVE SIX \n";
    op_file.close();

    ifstream ip_file("test.txt");

    char ch;
    while( 1 )
    {
        ip_file>>ch;
        if( ip_file.eof() )
            break;
        cout << ch ;        //display it
    }
}
```

OUTPUT

```
ONETWOTHREEFOURFIVESIX
```

Solution-1:

```
int main()
{
    ofstream op_file("test.txt");

    op_file<< "ONE TWO THREE \n";
    op_file<< "FOUR FIVE SIX \n";
    op_file.close();

    ifstream ip_file("test.txt");

    char ch;
    while( 1 )
    {
        ch = ip_file.get();
        if( ip_file.eof() )
            break;
        cout << ch ;      //display it
    }
}
```

Solution-2:

```
5   int main()
6   {
7       ofstream op_file("test.txt");
8       const int MAX = 80;
9       char buffer[MAX];
10
11      op_file<< "ONE TWO THREE \n";
12      op_file<< "FOUR FIVE SIX \n";
13      op_file.close();
14
15      ifstream ip_file("test.txt");
16      while( 1 )
17      {
18          ip_file.getline( buffer , MAX );
19          if( ip_file.eof() )
20              break;
21          cout << buffer <<"\n" ;
22      }
23  }
```

Example-3: File Output/Input

```
4   int main()
5   {
6       char ch1, ch2;
7       int  j1, j2;
8       double d1, d2;
9       string str1, str2;
10      char filename[] = "test.txt";
11      ch1 = 'A'; j1 = 12; d1 = 15.5; str1 = "Hello";
12      ofstream op_file( filename );
13      op_file << ch1 << ' ' << j1 << ' ' << d1 << ' ' << str1;
14      op_file.close();
15
16
17
18
19
20
21
22  }
```

Example-3: File Output/Input

```
4   int main()
5   {
6       char ch1, ch2;
7       int  j1, j2;
8       double d1, d2;
9       string str1, str2;
10      char filename[] = "test.txt";
11      ch1 = 'A'; j1 = 12; d1 = 15.5; str1 = "Hello";
12      ofstream op_file( filename );
13      op_file << ch1 << ' ' << j1 << ' ' << d1 << ' ' << str1;
14      op_file.close();
15
16      ifstream ip_file( filename );
17      while( !ip_file.eof() )
18      {
19          ip_file >> ch2 >> j2 >> d2 >> str2;
20          cout<< ch2 <<"\t"<< j2 <<"\t"<< d2 <<"\t"<< str2 <<endl;
21      }
22  }
```

File opening mode - ios::out

```
2  #include <fstream> //Required for file I/O
3  using namespace std;
4  int main()
5  {
6      ofstream myfile ("message.txt",ios::out );
7      if (!myfile)
8      {
9          //check if the file is opened or not
10         cout<<"\n Cannot open this file";
11         return 1;
12     }
13     myfile<<"mca\n";
14     myfile.close();                //close the file
15 }
```

File opening mode - ios::app

```
1  #include<iostream>
2  #include <fstream> //Required for file I/O
3  using namespace std;
4  int main()
5  {
6      ofstream myfile ("message.txt",ios::app );
7      if (!myfile)
8      {
9          //check if the file is opened or not
10         cout<<"\n Cannot open this file";
11         return 1;
12     }
13     myfile<<"mca\n";
14     myfile.close();                //close the file
15 }
```

File I/O using fstream object

```
4   int main()
5   {
6       fstream myfile;
7       char msg[75];
8       myfile.open( "message.txt" ,ios::app);
9       if (!myfile)
10      {                   //check if the file is opened or not
11          cout<<"\n Cannot open this file";
12          return 1;
13      }
14      cout<<"Enter a msg: "; cin.getline(msg,75);
15      myfile << msg << "\n";
16
17      myfile.close();                 //close the file
```

File I/O using fstream object…

```
19      myfile.open("message.txt", ios::in );
20
21      cout<<"\nThe file content is:\n";
22      while( 1 )
23      {
24          myfile.getline(msg,75);
25          if( myfile.eof() )
26              break;
27          cout << msg << endl;
28      }
29      myfile.close();  //close the file
30  }
```

Formatted vs unformatted file I/O

- The formatted IO converts numeric values (such as int, double) from their internal representations (e.g., 16-/32-bit int, 64-bit double) to a stream of characters that representing the numeric values in text form.

- The unformatted IO (e.g., write()) stores the bytes as they are, without format conversion.

Object I/O

```
4   class Student
5   {
6       char name[30];
7       int RNO;
8       float avg_marks;
9   public:
10      void getData()
11      {
12          cout<<"Enter name: ";
13          cin.getline(name,30);
14          cout<<"Roll No: "; cin>>RNO;
15          cout<<"Enter Avg_Marks: "; cin>>avg_marks;
16      }
17      void showData()
18      {
19          cout<<"Name:"<< name <<" , RNO:"<<RNO
20              <<" , Avg_Marks:"<< avg_marks;
21      }
22  };
```

```
23  int main()
24  {
25      Student S1,S2;
26      ofstream op_file("Student_Data.txt");
27      if( !op_file )
28      {
29        cout<<"Error in creating file.."<<endl;  return 0;
30      }
31
32      S1.getData();     //read from user
33      op_file.write( (char*) &S1 , sizeof(S1) );     //write into file
34
35      op_file.close();   //close the file
36
37      ifstream ip_file("Student_Data.txt");
38      ip_file.read( (char*) &S2, sizeof(S2) );
39
40      cout<<"\nThe contents of the file: \n";
41      S2.showData();
42  }
```

Sequential access vs Random access to a file

- Every file maintains two pointers called get_pointer (in input mode file) and put_pointer (in output mode file) which tells the current position in the file where reading or writing will takes place.
- In C++, random access is achieved by manipulating seekg(), seekp(), tellg() and tellp() functions.
- The seekg() and tellg() functions allow you to set and examine the get_pointer, and the seekp() and tellp() functions perform these operations on the put_pointer.

tellg(), tellp(), seekg(), seekp()

- **tellg()** returns the current position of the get_pointer
- **tellp()** returns the current position of the put_pointer
- **seekg() –** moves the get_pointer by specified number of bytes from the reference point

  Syntax: ifstream_obj **.** seekg (number_of_bytes , Reference_point);

  - The reference points are:

    ios::beg – from beginning of file
    ios::end – from end of file
    ios::cur – from current position in the file.

- **seekp() -** moves the put_pointer by specified number of bytes from the reference point

  Syntax: ofstream_obj **.** seekp (number_of_bytes , Reference_point);

examples

```
ifstream    fin;  ofstream   fout;
fin.seekg(30);   // will move the get_pointer (in ifstream) to byte number 30 in the file
 fout.seekp(30);  // will move the put_pointer (in ofstream) to byte number 30 in the file

fin.seekg(30,   ios::beg);   // go to byte no. 30 from beginning of file linked with fin
fin.seekg(-2,  ios::cur);    // back up 2 bytes from the current position of get pointer
fin.seekg(0,  ios::end);    // go to the end of the file
fin.seekg(-4,  ios::end)    // backup 4 bytes from the end of the file
```

Example: File pointer position

```
4   int main()
5   {
6       char ch = 'A';
7       fstream file( "Test.txt" ,  ios::out )
8       for( int i = 0; i < 10; i++, ch++ )
9           file << ch;
10      file.seekp( 2 );
11      file<<"Hello";
12      file.close();
13      file.open("Test.txt" ,  ios::in );
14      file.seekg( 2 );
15      while( 1 )
16      {
17          file.get( ch );
18          if( file.eof() )
19              break;
20          cout<<ch;
21      }
22  }
```

OUTPUT

```
HelloHIJ
```

Question-1:

```
4   int main()
5   {
6       char filename[] = "Test.txt";
7       char s[] = "ONE TWO THREE";
8       char ch;
9       int i = 0, pos;
10      fstream file;
11      file.open( filename, ios::out );
12
13      while( s[i] != '\0')
14      {
15          file.put( s[i] );
16          i++;
17      }
18      int len = file.tellp();
19      cout<<"\nLength of the file="<< len <<"\n";
20      file.close();
21      file.open( filename, ios::in );
22      file.seekg(0);
23      while( 1 )
24      {
25          pos = file.tellg();
26          cout << pos;
27          ch = file.get();
28          if( file.eof() )
29              break;
30          cout<<"\t"<< ch <<"\n";
31      }
32  }
```

OUTPUT

```
Length of the file=13
0        O
1        N
2        E
3
4        T
5        W
6        O
7
8        T
9        H
10       R
11       E
12       E
13
```

Question-2:

```
class Emp
{
    int emp_id;
    int age;
    double salary;
public:
    Emp(){}

    Emp( int id, int age, double sal )
    {
        emp_id = id; this->age = age;
        salary = sal;
    }
    void show()
    {   cout<<emp_id <<"\t"
            <<age<<"\t" <<salary;
    }
};
```

```
23  int main()
24  {
25      char filename[] = "Emp_info.txt";
26      fstream file;
27      file.open(filename, ios::out|ios::binary);
28
29      Emp P[] = { {0, 25, 23000 }, {1, 29, 55000}, {2,26,54300},
30                  {3, 34, 75000 }, {4, 30,60000 }};
31
32      file.write((char*)P, sizeof(P));
33      file.close();
34
35      file.open( filename, ios::in |ios::binary );
36      Emp emp_obj;
37      cout<<"\n size of employee obj: "<<sizeof(emp_obj);
38
39      file.seekg(0,ios::end);
40      int cur_loc_getptr = file.tellg();
41      cout<<"\nThe current location of getptr: "<<cur_loc_getptr;
42      int total_objects = cur_loc_getptr / sizeof( emp_obj );
```

```
45      int emp_id;
46      cout<<"\n\nEnter the id of the employee to be searched:";
47      cin >> emp_id;
48      long location = emp_id * sizeof( emp_obj );
49      file.seekg( location );
50      file.read( (char*) &emp_obj, sizeof(emp_obj));
51      file.close();
52      emp_obj.show();
53  }
```

OUTPUT

```
 size of employee obj: 16
The current location of getptr: 80
 The total No. of objects: 5

Enter the id of the employee to be searched: 3
3        34       75000
-----------------------------------
```

Cin.ignore()

- **Syntax:** cin.ignore(int n = 1, int delim = EOF);

- Extracts characters from the input sequence and discards them, until either *n* characters have been extracted, or one compares equal to *delim*.

Overloading the extraction(>>) and insertion(<<) operator

Overloading the extraction(>>) and insertion(<<) operator

```
friend ostream & operator << ( ostream &out, class_type   obj )
{
        // statements..
        return out;
}
```

- The first parameter to the function is a reference to the output stream. The second parameter is the object being inserted.

```
friend istream & operator >> ( istream &in, class_type   obj )
{
        // statements..
        return in;
}
```

```
3   class Complex
4   {
5   private:
6       int real, imag;
7   public:
8       Complex(int r = 0, int i =0)
9       {  real = r;    imag = i; }
10
11      friend ostream & operator << (ostream &, Complex &);
12      friend istream & operator >> (istream &, Complex &);
13  };
```

```
ostream & operator << (ostream &out,  Complex &C)
{
    out << C.real;
    out << "+i" << C.imag << endl;
    return out;
}

istream & operator >> (istream &in,  Complex &C)
{
    cout << "Enter Real Part ";
    in >> C.real;
    cout << "Enter Imaginary Part ";
    in >> C.imag;
    return in;
}
```

```
 Enter 2 complex numbers:

Enter Real Part 2
Enter Imaginary Part 1

Enter Real Part 3
Enter Imaginary Part 4
The complex numbers are:
2+i1

3+i4
```

```
int main()
{
    Complex c1,c2;
    cout<<"\n Enter 2 complex numbers: \n";
    cin >>c1>>c2;
    cout << "The complex numbers are: \n";
    cout << c1<<"\n"<<c2;
}
```

Manipulators

- Manipulators are operators used in C++ for formatting output.

  — endl

  — setw

  — setfill

  — setbase

  — setprecision

  ……

- **setw**: manipulator sets the width of the field assigned for the output.

Syntax: Setw(n) , n –> Number of characters to be used as field width.

```
4   int main()
5   {
6       int a=7, b=77, c=777;
7       cout <<setw(10);
8       cout <<a;
9       cout <<"\n"<<setw(10)<<b<<"\n"<<setw(10)<<c;
10      return 0;
11  }
```

```
         7
        77
       777
```

- **setfill** character is used to fill spaces when results have to be padded to the field width.

```
int a=7, b=77, c=777;

cout <<setfill ('x') << setw(10);
cout <<a;

cout <<"\n"<< setfill ('*') << setw(10) << b <<"\n";

cout<< setfill ('#' ) << setw(10) << c;
```

```
xxxxxxxxx7
********77
#######777
```

- **setprecision**

✓The setprecision Manipulator is used to set the number of digits printed to the right of the decimal point.

✓This may be used in two forms:

- ✓fixed
- ✓scientific

```
float x = 3.142857;
cout << fixed << setprecision(5) << x << endl;
cout << fixed << setprecision(3) << x << endl;
cout << scientific <<setprecision(3) <<x << endl;
```

```
3.14286
3.143
3.143e+000
```

- setbase (int base);

decimal : if base is 10
hexadcimal : if base is 16
octal : if base is 8

```
ff
377
255
```

```
int x = 255;
// set base to hexadecimal
cout << setbase(16);

// displaying 255 in hexadecimal
cout << x <<endl;

// set base to Octal
cout << setbase(8);

// displaying 255 in Octal
cout << x << endl;

// displaying 255 in decimal
cout<< setbase(10);
cout<<x;
```