Marwadi
education foundation

Prof. Meghnesh Jayswal

UNIT 7
Streams

Objective of this Unit

- Concept of Stream
- C++ Stream Classes
- Formatted and unformatted I/O operations
- Manipulators

Concept Of stream

Managing console I/O operation

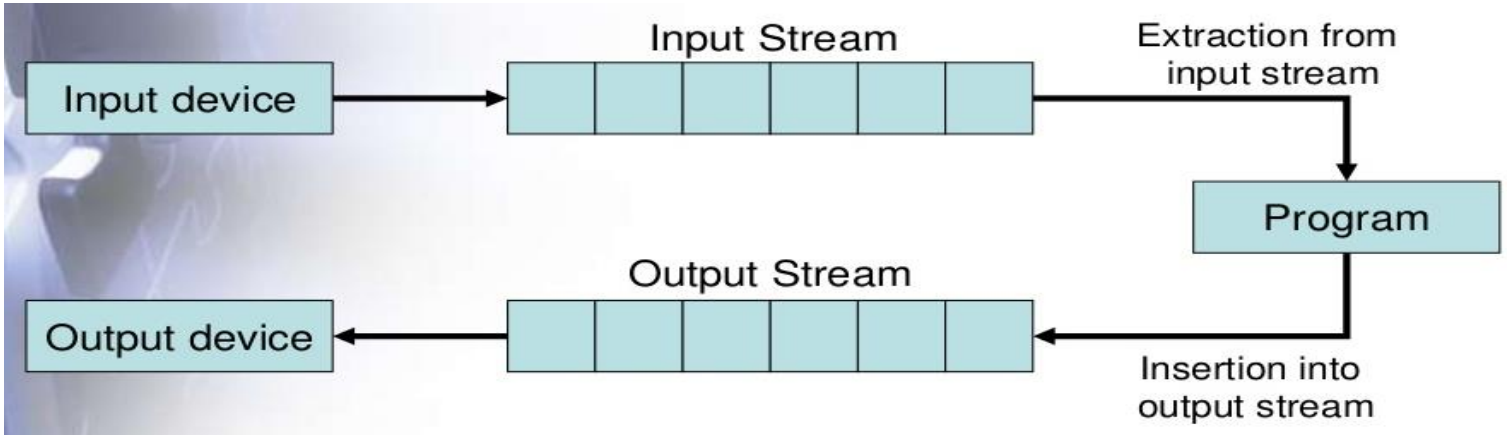
- C++ uses the concept of stream and stream classes to implement its I/O operations with the console and disk files.

- C++ supports all of C's rich set of I/O functions.

- **Stream is a general name given to a flow of data.**

C++ Stream

- Stream is an interface supplied by the I/O system of C++ between the programmer and the actual device being accessed.
- It will work with devices like terminals, disks and tape drives.
- **A stream is a sequence of bytes.**
- It acts either as a source from which the input data can be obtained or as a destination to which the output data can be sent.

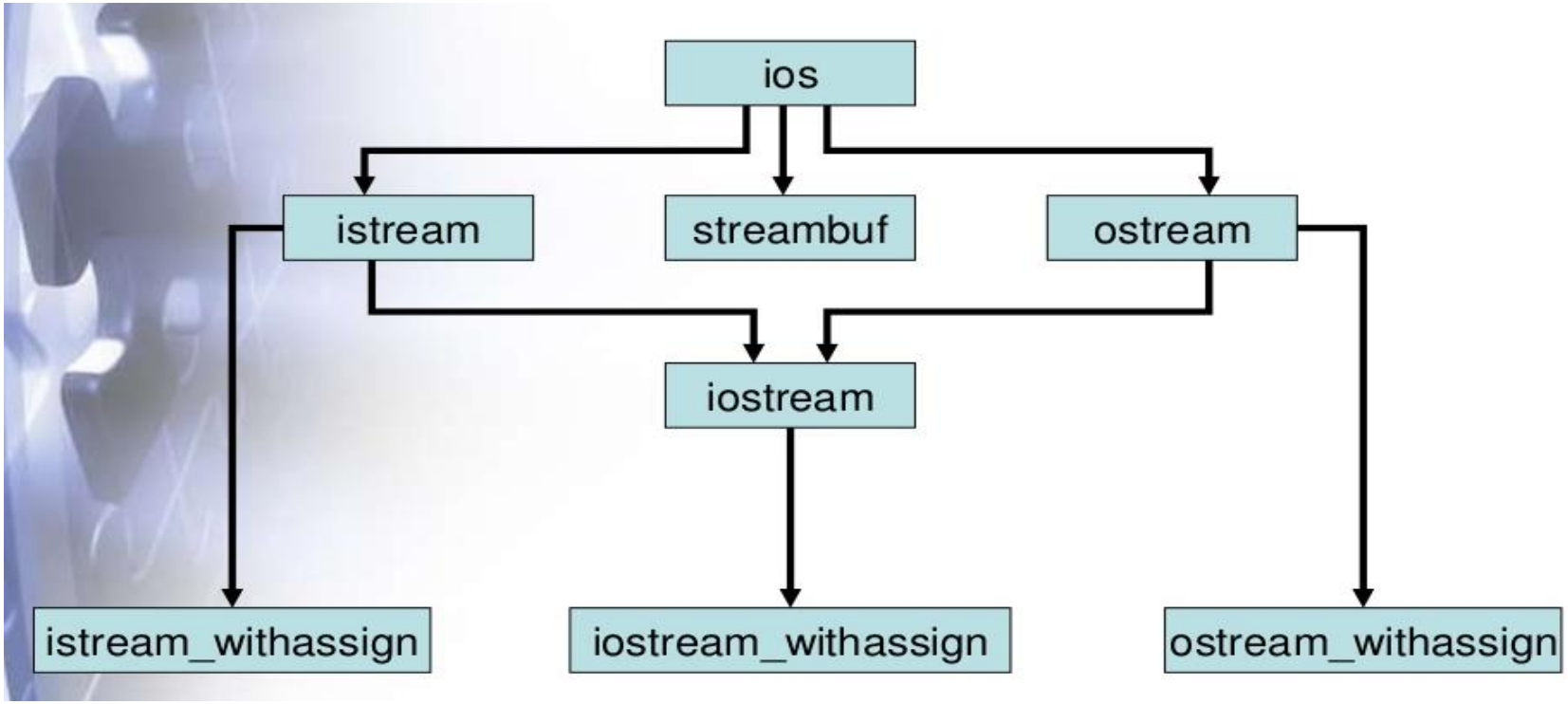


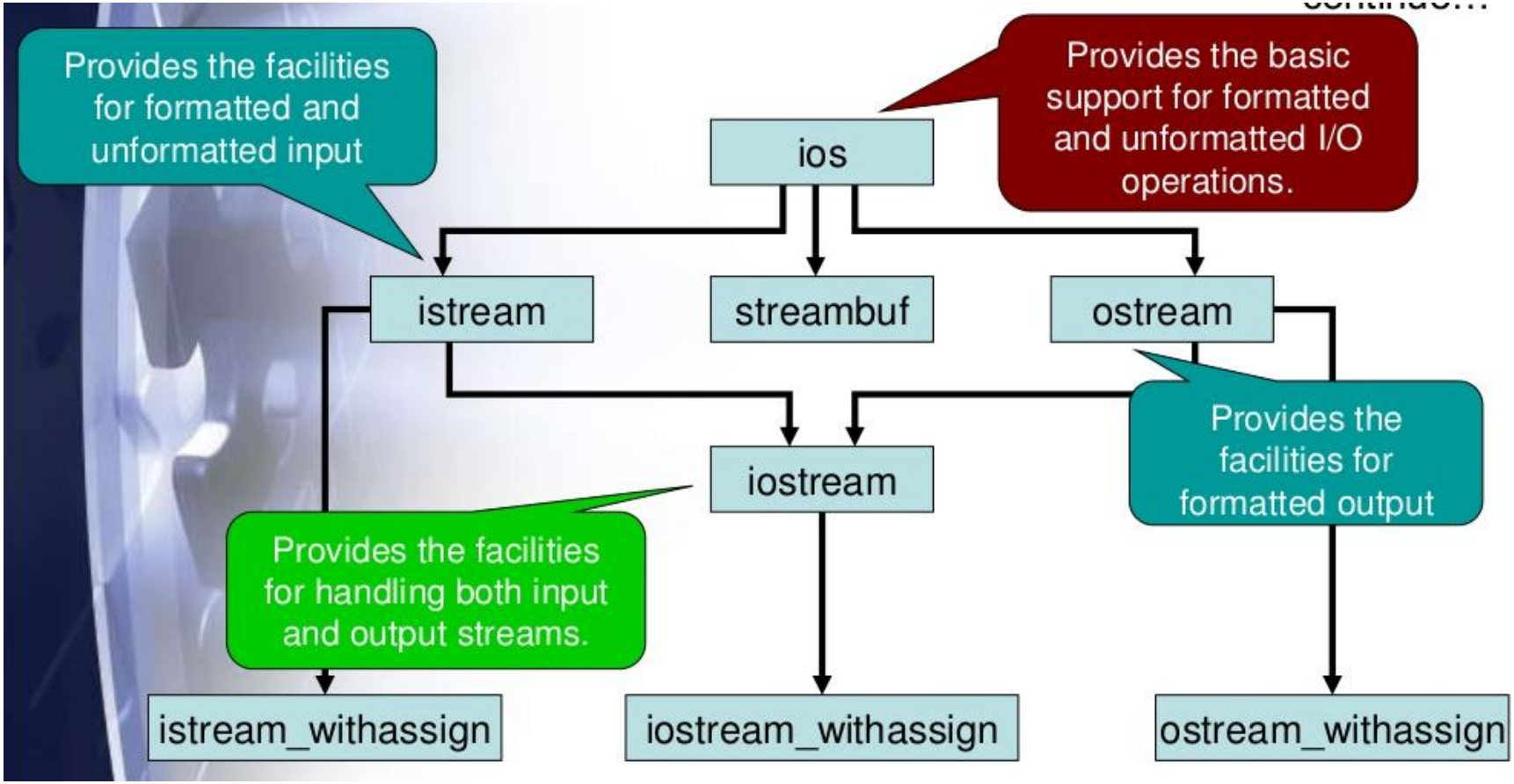
- **Input Stream:-** The Source that provides data to the program.
- **Output Stream:-**The destination streams that receives output from the program.

- The data in the input stream can come **from keyboard** or any other storage device.
- The data in the output stream can go to the **screen** or any other **storage devices.**

C++ stream Classes

- The C++ I/O system contains a hierarchy of classes that are used to define various streams to deal with both the console and disk files.
- These classes are called stream classes.
- These classes are declared in the header file iostream.




Provides the facilities for formatted and unformatted input
Provides the basic support for formatted and unformatted I/O operations.
ios
istream
streambuf
ostream
iostream
Provides the facilities for formatted output
Provides the facilities for handling both input and output streams.
istream_withassign
iostream_withassign
ostream_withassign

Unformatted I/O Operations

Overloaded Operators >> and <<

- The objects cin and cout are used for input and output of data of various types.

By overloading the operators >> and <<

- >> operator is overloaded in the istream class.
- << operator is overloaded in the ostream class.
- This is used for input data through keyboard.
- cin>>var1>>var2…..>>varN

  (where var1,var2,..,varN are valid variable name)

- cout<<item1<<item2….<<itemN

  (where item1,item2….itemN may be variables or constants of a basic type)

Unformatted I/O Operations

- **put() and get() Functions**
- get() and put() are member functions of istream and ostream classes.
- For single character input/output operations.
- There two types of get() functions:
  - get(char *) :-Assigns the input character to its argument
  - get(void) :-Returns the input character

```
char c;
cin.get(c);
c=cin.get();
```

- put():-used to output a line of text, character by character.

```
char c;
cout.put('x');
cout.put(c);
```

Unformatted I/O Operations

- **getline() and write() functions**
- **getline()** function reads a whole line of text that ends with a newline character. The new line character is read but not saved.
- cin.getline(line,size);

- Reading is terminated as soon as either the newline character '\n' is encountered or size-1 characters are read.

- **write()** function displays an entire line of text.
- cout.write(line,size);
- write() also used to concatenate strings.

Formatted console I/O Operations

- C++ supports a number of features that could be used for formatting the output,

- These features include:
  - ios class functions and flags.
  - Manipulators.
  - User-defined output functions.

ios member functions

- **width() :-** To specify the required field size for displaying an output value.
- **precision() :-** To specify the number of digits to be displayed after the decimal point of a float value.
- **fill() :-**To specify a character that is used to fill the unused portion of a field.
- **setf() :-**To specify format flags that can control the form of output display.
- **unsetf() :-**To clear the flags specified.

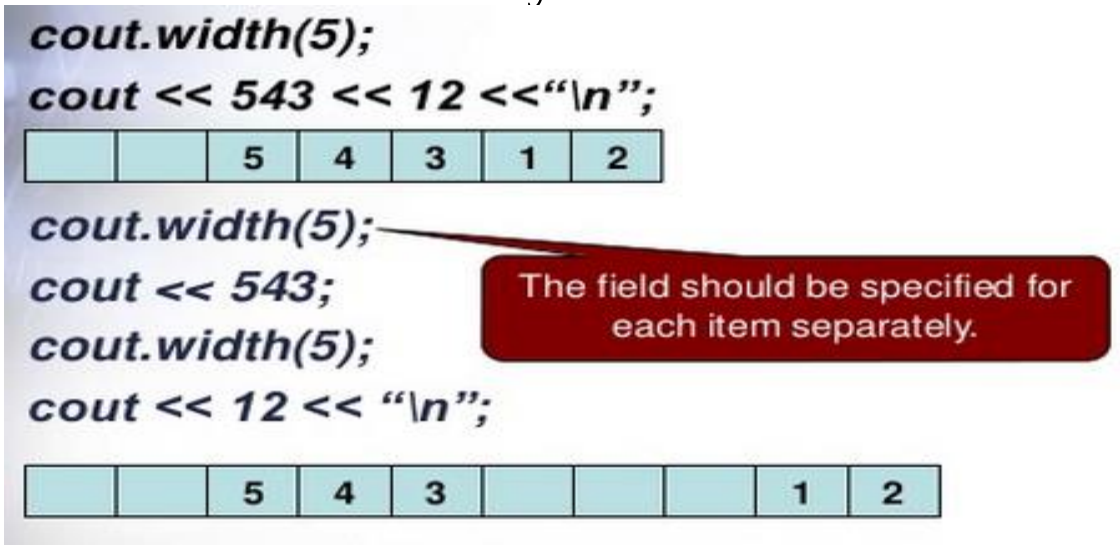
Manipulators

- **Manipulators** are special functions that can be included in the I/O statement to alter the format parameters of a stream.
- To access manipulators, the file **iomanip.h** should be included in the program.
  - setw()
  - setprecision()
  - setfill()
  - setiosflags()
  - resetiosflags()

width() :-Defining Field width

- To define the width of a field necessary for the output of an item.
- Since it is a member function, we have to use an object to invoke it.

  cout.width(w) ,Where w is the field width(number of columns).
- The output will be printed in a field of w character wide at the right end of the field.
- The width() function can specify the field width for only one item-item that follows immediately.



precision() :-Setting Precision

- Used to specify the number of digits to be displayed after the decimal point while printing the floating-point numbers.
- By default, the floating numbers are printed six digits after the decimal point.

- cout.precision(d);
  - *Where d is the number of digits to the right of the decimal point.*

```
cout.precision(3);
cout << sqrt(2) << endl;
cout << 3.14159 << endl;
cout <<2.50032 << endl;
```

1.141	→ truncated
3.142	→ rounded
2.5	→o trailing zeros

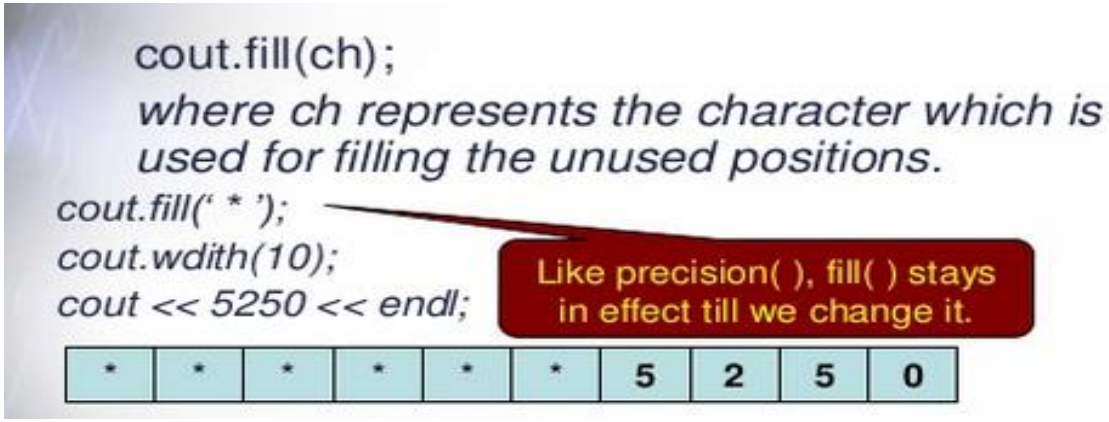
precision() :-Setting Precision

- Unlike width(), precision() retains the setting in effect until it is reset.
- We can also combine the field specification with the precision setting.

```
cout.precision(2);
cout.width(5);
cout << 1.2345;
```

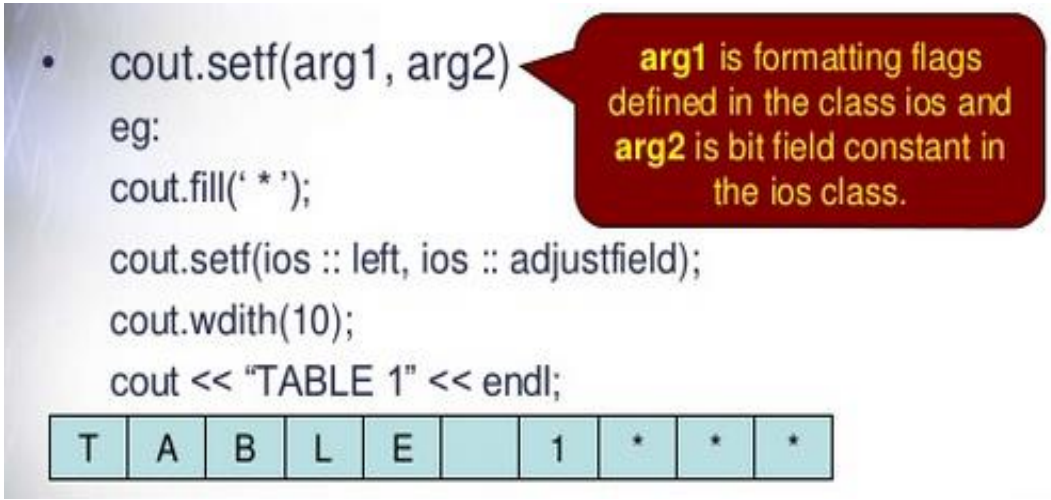
fill() :-Filling and padding

- When printing values with larger field width then required by the values, the unused positions of the field are filled with white spaces, by default.
- fill() function can be used to fill the unused positions by any desired character.

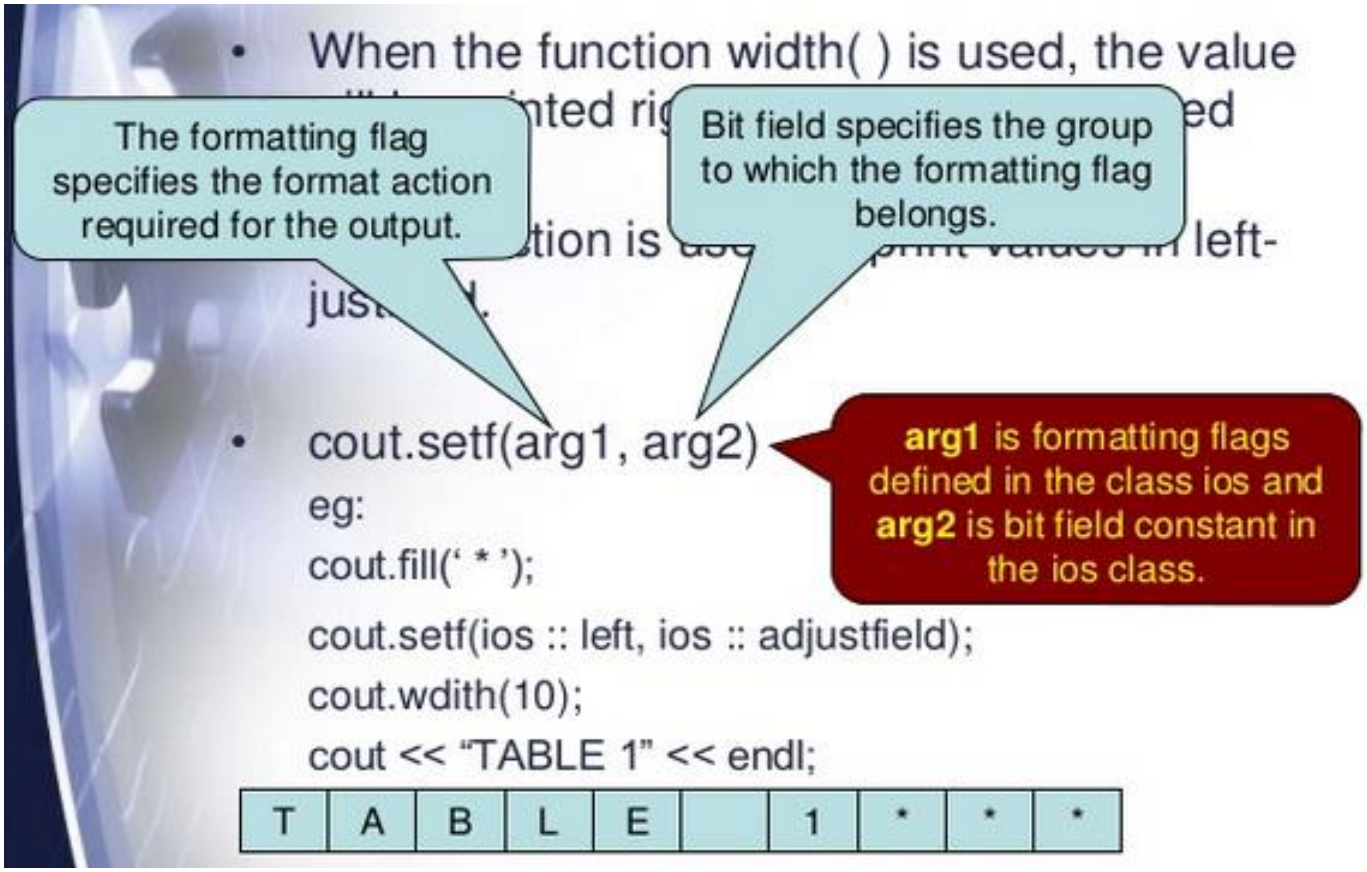


setf() :-formatting flags, bit-fields

- When the function width() is used, the value will be printed right-justified in the created width.
- setf() function is used to print values in left-justified.



setf() :-formatting flags, bit-fields



Managing output with Manipulators

- The header file iomanip provides a set of functions called manipulators which can be used to manipulate the output formats.
- Some manipulators are more convenient to use the member functions and flags of ios.
- Two or more manipulators can be used as a chain in one statement.

```
cout<<manip1<<manip2<<manip3<<item;
cout<<manip1<<item1<<manip2<<item2;
```

Managing output with manipulators

- Manipulators and their meanings

Manipulators	Meaning	Equivalent
setw(int w)	Set the field width to w	width( )
setprecision(int d)	Set floating point precision to d	precision( )
setfill(int c)	Set the fill character to c	fill( )

```
cout << setw(5) << setprecision(2) << 1.2345
     << setw(10) << setprecision(4) << sqrt(2);
```

- We can jointly use the manipulators and the ios functions in a program.

Manipulators & ios member function

- The ios member function return the previous format state which can be used later.
- But the manipulator does not return the previous format state.

```
cout.precision(2);  // previous state.
int p =cout.precision(4); // current state, p=2.
cout.precision(p); // change to previous state
```

Designing our own Manipulators

- We can design our own manipulators for certain special purpose.

```
ostream & manipulator ( ostream & output)
{
      ..... (code)
      return output;
}
ostream & unit (ostream & output)
{
      output << "inches";
      return output;
}
cout << 36 << unit;          → will produce "36 inches".
```

Designing our own Manipulators

- We can also create manipulators that could represent a sequence of operations:

```
ostream & show (ostream & output)
{
  output.setf(ios::showpoint);
  output.setf(ios::showpos);
  output << setw(10);
  retum output;
}
```

- This function defines a manipulator called show that turns on the flags showpoint and showpos declared in the class ios and sets the field width to 10.

Thank You