```
Pr
P
C++ Programming
C++ Progr
```

The last line of the output indicates that the statement

```
cout.write(string1,10);
```

displays more character than what is contained in string1.

It is possible to concatenate two strings using the write() function.The statement

```
cout.write(string1,m).write(string2,n);
```

is equivalent to the following two statements:

```
 cout.write(string1,m);
cout.write(string2,n);
```

5.5 Formatted Console I/O Operations:-

C++ supports a number of features that could be used for formatting the output.These features include:

--ios class function and flags.

--manipulators.

--User-defined output functions.

The ios class contains a large number of member functions that would help us to format the output in a number of ways.The most importany ones among them are listed in table 5.2

Table 5.2 ios format functions

Function	**Task**
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 float value
Fill()	To specify a character that is used to fill the unused portion of a field

	a field
Setf()	To specify format flags that can control the form of output display(such as left-justification and right-justification)
Unsetf()	To clear the flags specified

Manipulators are special functions that can be included in the I/O statements to alter the format parameter of stream .Table 5.3 shows some important manipulator functions that are frequently used. To access these manipulators, the file iomanip should be included in the program.

Table 5.3 Manipulators

Manipuators	**Equivalent ios function**
setw()	width()
setprecision()	precision()
setfill()	fill()
setiosflags()	setf()
resetiosflags()	unsetf()

In addition to these standard library manipulators we can create our own manipulator functions to provide any special output formats.

5.5.1 Defining Field Width:width()

The width() function is used to define the width of a field necessary for the output of an item.As it is a member function object is required to invoke it like

```
cout.width(w);
```

here w is the field width.The output will be printed in a field of w character wide at the right end of field.The width() function can specify the field width for only one item(the item that follows immediately).After printing one item(as per the specification) it will revert back the default.for example,the statements

```
cout.width(5);

cout<<543<<12<<"\n";
```

will produce the following output:

		5	4	3	1	2

The value 543 is printed right justified in the first five columns.The specification width(5) does not retain the setting for printing the number 12.this can be improved as follows:

```
  cout.width(5);

  cout<<543;

 cout.width(5);

cout<<12<<"\n";
```

This produces the following output:

		5	4	3				1	2

The field width should be specified for each item.C++ never truncate the values and therefore,if the specified field width is smaller than the size of the value to be printed,C++ expands the field to fit the value.program 5.4 demonstrates how the function width() works.

Program 5.4

Specifying field size with width()

```
#include <iostream>

using namespace std;

int main()

{

 int item[4] ={ 10,8,12,15};

int cost[4]={75,100,60,99};

cout.width(5);

cout<<"Items";

cout.width(8);

cout<<"Cost";

cout.width(15);
cout<<"Total Value"<<"\n";
int  sum=0;
for(int i=0;i<4 ;i++)
{
  cout.width(5);
cout<<items[i];
cout.width(8);
cout<<cost[i];
int value = items[i]  * cost[i];
cout.width(15);
cout<<value<<"\n";
sum= sum + value;


}
cout<<"\n Grand total = ";
cout.width(2);
cout<<sum<<"\n";
return 0;
}
```

The output of program 5.4 would be

```
ITEMS                  COST          TOTAL VALUE
10                        75             750
8                        100              800
12                       60               720
15                       99               1485
Grand total  =3755
```

5.5.2 Setting Precision: precision():-

By default ,the floating numbers are printed with six digits after the decimal points. However ,we can specify the number of digits to be displayed after the decimal point while printing the floating point numbers.

This can be done by using the precision () member function as follows:

```
cout.precision(d);
```

where d is the number of digits to the right of decimal point.for example the statements

```
cout.precision(3);

cout<<sqrt(2)<<”\n”;

cout<<3.14159<<”\n”;

cout<<2.50032<<”\n”;
```

will produce the following output:

```
1.141          (truncated)

3.142           (rounded to nearest cent)

2.5               (no trailing zeros)
```

Unlike the function width(),precision() retains the setting in effect until it is reset.That is why we have declared only one statement for precision setting which is used by all the three outputs.We can set different valus to different precision as follows:

```
cout.precision(3);

cout<<sqrt(2)<<”\n”;

cout.precision(5);

cout<<3.14159<<”\n”;
```

We can also combine the field specification with the precision setting.example:

```
cout.precision(2);

cout.width(5);

cout<<1.2345;
```

The first two statement instruct :”print two digits after the decimal point in a field of five character width”.Thus the output will be:

	1		2	3

Program 5.5 shows how the function width() and precision() are jointly used to control the output format.

Program 5.5

PRECISION SETTING WITH precision()

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
cout<<"precision set to 3 digits\n\n";
cout.precision(3);
cout.width(10);
cout<<"value";
cout.width(15);
cout<<"sqrt_of _value"<<"\n";
for (int n=1;n<=5;n++)
{
   cout.width(8);
cout<<n;
cout.width(13);
cout<<sqrt(n)<<"\n";
}
cout<<"\n precision set to 5 digits\n\n";
cout.precision(5);
cout<<"sqrt(10) = " <<sqrt(10)<<"\n\n";
cout.precision(0);
cout<<"sqrt(10) = " <<sqrt(10)<<"(default setting)\n";
return 0;
}
```

The output is

Precision set to 3 digits

VALUE	SQRT OF VALUE
1	1
2	1.41
3	1.73
4	2
5	2.24

Precision set to 5 digits

Sqrt(10)=3.1623

Sqrt(10)=3.162278 (Default setting)

<u>5.5.3 FILLING AND PADDING :fill()</u>

The unused portion of field width are filled with white spaces, by default. The fill() function can be used to fill the unused positions by any desired character.It is used in the following form:

cout.fill(ch);

Where ch represents the character which is used for filling the unused positions.Example:

cout.fill('*');

cout.width(10);

cout<<5250<<"\n";

The output would be:

*	*	*	*	*	*	5	2	5	0

Financial institutions and banks use this kind of padding while printing cheques so that no one can change the amount easily.Like precision (),fill()

Stays in effect till we change it.As shown in following program

Program 5.6

```
#include<iostream>
using namespace std;
int main()
{  cout.fill('<');
cout.precision(3);
for(int  n=1;n<=6;n++)
{
 cout.width(5);
cout<<n;
cout.width(10);
cout<<1.0/float(n)<<"\n";
if(n==3)
cout.fill('>');
}
cout<<"\nPadding changed \n\n";
cout.fill('#');  //fill() reset
cout.width(15);
cout<<12.345678<<"\n";
return 0;
}
```

The output will be

```
<<<<1<<<<<<<<<<1

<<<<2<<<<<<<<0.5

<<<<3<<<<<<0.333

>>>>4>>>>>>0.25

>>>>5>>>>>>>0.2

PADDING CHANGED

#########12.346
```

<u>5.5.4 FORMATTING FLAGS,Bit Fields and setf():-</u>

The setf() a member function of the ios class, can provide answers left justified.The setf() function can be used as follows:

cout.setf(arg1.arg2)

The arg1 is one of the formatting flags defined in the class ios.The formatting flag specifies the format action required for the output.Another ios constant,arg2,known as bit field specifies the group to which the formatting flag belongs. for example:

cout.setf(ios::left,ios::adjustfield);

cout.setf(ios::scientific,ios::floatfield);

Note that the first argument should be one of the group member of second argument.

Consider the following segment of code:

```
cout.fill('*');

cout.setf(ios::left,ios::adjustfield);

cout.width(15);

cout<<"table1"<<"\n";
```

This will produce the following output:

T	A	B	L	E		1	*	*	*	*	*	*	*	*

The statements

```
cout.fill('*');

cout.precision(3);
```

cout.setf(ios::internal,ios::adjustfield);

cout.setf(ios::scientific,ios::floatfield);

cout.width(15);

cout<<-12.34567<<"\n";

Will produce the following output:

-	*	*	*	*	*	1	.	2	3	5	e	+	0	1

5.6 Managing Output with Manipulators:-

The header file iomanip provides a set of functions called manipulators which can be used to manipulate the output format. They provide the same features as that of the ios member functions and flags.For example,two or more manipulators can be used as a chain in one statement as follows

cout<<manip1<<manip2<<manip3<<item;

cout<<manip1<<item1<<manip2<<item2;

This kind of concatenation is useful when we want to display several columns of output. The most commonly used manipulators are shown below

In the table 5.4

Manipulator	**Meaning**	**Equivalent**
setw(int w)	Set the field width to w	width()
setprecision(int d)	Set the floating point precision to d	precision()
setfill(int c)	Set the fill character to c	fill()
setiosflags(long f)	Set the format flag f	setf()
resetiosflags(long f)	Clear the flag specified by f	unsetf()
endl	Insert new line and flush stream	"\n"

Examples of manipulators are given below:

cout<<setw(10)<<12345;