FAST

الذی علم بالقلم۔ علم الانسان ما لم یعلم۔

National University of Computer and Emerging Sciences

DEPARMENT OF COMPUTER SCIENCE

Object Oriented Programming-Lab

Lab-07

Engr. Khuram Shahzad, FAST NUCES Peshawar

OOP (LAB-07)
MANIPULATOR & FILE INPUT/OUTPUT

Engr. Khuram Shahzad
(Instructor)

LAB-07 CONTENTS

- **Manipulator** (fixed & scientific, showpoint, setprecision, , setw, setfill and Left and right Manipulators)
- Debugging: Understanding Logic Errors and Debugging with cout Statements
- **File Input/Output (Open, Read, Write the file)**
- **Programming Example:** Movie Tickets Sale and Donation to Charity
- **Programming Example:** Student Grade

MANIPULATOR

Fixed & Scientific Manipulator

- To output floating-point numbers in a fixed decimal format, you use the manipulator fixed. The following statement sets the output of floating-point numbers in a fixed decimal format on the standard output device:
- **cout << fixed;**

- After the preceding statement executes, all floating-point numbers are displayed in the fixed decimal format until the manipulator fixed is disabled. You can disable the manipulator fixed by using the stream member function unsetf. For example, to disable the manipulator fixed on the standard output device, you use the following statement:
- **cout.unsetf(ios::fixed);**

- The following example shows how the manipulators scientific and fixed work without using the manipulator setprecision.

- Note: On some compliers, the statements
- **cin >> fixed; and cin >> scientific;** might not work.
- In this case, you can use **cin.setf(ios::fixed); in the place of cin >> fixed;** and **cin.setf(ios::scientific); in place of cin >> scientific;**

```cpp
#include <iostream>
using namespace std;
int main() {
    double hours = 35.45;
    double rate = 15.00;
    double tolerance = 0.01000;
    cout << "hours = " << hours << ", rate = " << rate
         << ", pay = " << hours * rate
         << ", tolerance = " << tolerance << endl << endl;
    cout << scientific;
    cout << "Scientific notation: " << endl;
    cout << "hours = " << hours << ", rate = " << rate
         << ", pay = " << hours * rate
         << ", tolerance = " << tolerance << endl << endl;
    cout << fixed;
    cout << "Fixed decimal notation: " << endl;
    cout << "hours = " << hours << ", rate = " << rate
         << ", pay = " << hours * rate
         << ", tolerance = " << tolerance << endl << endl;
    return 0;
}
```

```
D:\Object Oriented Language\OOP Lab-07\OOP Lab-07 Programs\fixed.exe
hours = 35.45, rate = 15, pay = 531.75, tolerance = 0.01

Scientific notation:
hours = 3.545000e+001, rate = 1.500000e+001, pay = 5.317500e+002, tolerance = 1.000000e-002

Fixed decimal notation:
hours = 35.450000, rate = 15.000000, pay = 531.750000, tolerance = 0.010000
```

Fixed, Showpoint & setprecision Manipulator

- Suppose that the decimal part of a decimal number is zero. In this case, when you instruct the computer to output the decimal number in a fixed decimal format, the output may not show the decimal point and the decimal part. To force the output to show the decimal point and trailing zeros, you use the manipulator **showpoint**. The following statement sets the output of decimal numbers with a decimal point and trailing zeros on the standard input device:
- **cout << showpoint;**

- Of course, the following statement sets the output of a floating-point number in a fixed decimal format with the decimal point and trailing zeros on the standard output device: **cout << fixed << showpoint;**
- The program in Example illustrates how to use the **manipulators setprecision, fixed, and showpoint**

fixed.cpp SetprecisionFixedShowpoint.cpp

```
1  //Example: setprecision, fixed, showpoint
2  #include <iostream> //Line 1
3  #include <iomanip> //Line 2
4  using namespace std; //Line 3
5  const double PI = 3.14159265; //Line 4
6  int main() { //Line 5
7      //Line 6
8      double radius = 12.67; //Line 7
9      double height = 12.00; //Line 8
10     cout << fixed << showpoint; //Line 9
11     cout << setprecision(2)
12          << "Line 10: setprecision(2)" << endl; //Line 10
13     cout << "Line 11: radius = " << radius << endl; //Line 11
14     cout << "Line 12: height = " << height << endl; //Line 12
15     cout << "Line 13: volume = "
16          << PI * radius * radius * height << endl; //Line 13
17     cout << "Line 14: PI = " << PI << endl << endl; //Line 14
18     cout << setprecision(3)
19          << "Line 15: setprecision(3)" << endl; //Line 15
20     cout << "Line 16: radius = " << radius << endl; //Line 16
21     cout << "Line 17: height = " << height << endl; //Line 17
22     cout << "Line 18: volume = "
23          << PI * radius * radius * height << endl; //Line 18
24     cout << "Line 19: PI = " << PI << endl << endl; //Line 19
25     cout << setprecision(4)
26          << "Line 20: setprecision(4)" << endl; //Line 20
27     cout << "Line 21: radius = " << radius << endl; //Line 21
28     cout << "Line 22: height = " << height << endl; //Line 22
29     cout << "Line 23: volume = "
30                  << PI * radius * radius * height << endl; //Line 23
31     cout << "Line 24: PI = " << PI << endl << endl; //Line 24
32     cout << "Line 25: "
33          << setprecision(3) << radius << ", "
34          << setprecision(2) << height << ", "
35          << setprecision(5) << PI << endl; //Line 25
36     return 0; //Line 26
37 } //Line 2
```

D:\Object Oriented Language\OOP Lab-07\OOP Lab-07 Programs\SetprecisionFixedShowpoint.exe

```
Line 10: setprecision(2)
Line 11: radius = 12.67
Line 12: height = 12.00
Line 13: volume = 6051.80
Line 14: PI = 3.14

Line 15: setprecision(3)
Line 16: radius = 12.670
Line 17: height = 12.000
Line 18: volume = 6051.797
Line 19: PI = 3.142

Line 20: setprecision(4)
Line 21: radius = 12.6700
Line 22: height = 12.0000
Line 23: volume = 6051.7969
Line 24: PI = 3.1416

Line 25: 12.670, 12.00, 3.14159

--------------------------------
Process exited after 0.07017 seconds with return value
Press any key to continue . . .
```

setw Manipulator

- The manipulator **setw** is used to output the value of an expression in a specific number of columns. The value of the expression can be either a string or a number. The expression setw(n) outputs the value of the next expression in n columns. **The output is rightjustified.**
- Thus, if you specify the number of columns to be 8, for example, and the output requires only four columns, the first four columns are left blank.
- Furthermore, if the number of columns specified is less than the number of columns required by the output, the output automatically expands to the required number of columns; the output is not truncated.
- For example, if x is an int variable, the following statement outputs the value of x in five columns on the standard output device:
- **cout << setw(5) << x << endl;** To use the manipulator setw, the program must include the header file **iomanip**. Thus, the following include statement is required: #include <iomanip>
- Unlike **setprecision**, which controls the output of all floating-point numbers until it is reset, **setw** controls the output of only the next expression.

fixed.cpp | SetprecisionFixedShowpoint.cpp | setw.cpp

```
1  //Example: setw
2  #include <iostream>
3  #include <iomanip>
4  using namespace std;
5  int main() {
6      int x = 19; //Line 1
7      int a = 345; //Line 2
8      double y = 76.384; //Line 3
9      cout << fixed << showpoint; //Line 4
10     cout << "12345678901234567890" << endl; //Line 5
11     cout << setw(5) << x << endl; //Line 6
12     cout << setw(5) << a << setw(5) << "Hi"
13          << setw(5) << x << endl << endl; //Line 7
14     cout << setprecision(2); //Line 8
15     cout << setw(6) << a << setw(6) << y
16          << setw(6) << x << endl; //Line 9
17     cout << setw(6) << x << setw(6) << a
18          << setw(6) << y << endl << endl; //Line 10
19     cout << setw(5) << a << x << endl; //Line 11
20     cout << setw(2) << a << setw(4) << x << endl; //Line
21     return 0;
22 }
```

```
D:\Object Oriented Language\OOP Lab-07\OOP Lab-07 Programs\setw.exe
12345678901234567890
   19
  345   Hi   19

  345 76.38    19
   19   345 76.38

  34519
345  19

--------------------------------
Process exited after 0.1382 seconds w
Press any key to continue . . .
```

setw Manipulator

- The manipulator **setw** is used to output the value of an expression in a specific number of columns. The value of the expression can be either a string or a number. The expression setw(n) outputs the value of the next expression in n columns. **The output is rightjustified**. Thus, if you specify the number of columns to be 8, for example, and the output requires only four columns, the first four columns are left blank.
- Furthermore, if the number of columns specified is less than the number of columns required by the output, the output automatically expands to the required number of columns; the output is not truncated.
- For example, if x is an int variable, the following statement outputs the value of x in five columns on the standard output device:
- **cout << setw(5) << x << endl;** To use the manipulator setw, the program must include the header file iomanip. Thus, the following include statement is required: #include <iomanip>
- Unlike setprecision, which controls the output of all floating-point numbers until it is reset, setw controls the output of only the next expression.

fixed.cpp | SetprecisionFixedShowpoint.cpp | setw.cpp

```
1  //Example: setw
2  #include <iostream>
3  #include <iomanip>
4  using namespace std;
5  int main() {
6      int x = 19; //Line 1
7      int a = 345; //Line 2
8      double y = 76.384; //Line 3
9      cout << fixed << showpoint; //Line 4
10     cout << "12345678901234567890" << endl; //Line 5
11     cout << setw(5) << x << endl; //Line 6
12     cout << setw(5) << a << setw(5) << "Hi"
13          << setw(5) << x << endl << endl; //Line 7
14     cout << setprecision(2); //Line 8
15     cout << setw(6) << a << setw(6) << y
16          << setw(6) << x << endl; //Line 9
17     cout << setw(6) << x << setw(6) << a
18          << setw(6) << y << endl << endl; //Line 10
19     cout << setw(5) << a << x << endl; //Line 11
20     cout << setw(2) << a << setw(4) << x << endl; //Line
21     return 0;
22 }
```

```
D:\Object Oriented Language\OOP Lab-07\OOP Lab-07 Programs\setw.exe
12345678901234567890
   19
  345   Hi   19

  345 76.38    19
   19   345 76.38

  34519
345  19

--------------------------------
Process exited after 0.1382 seconds w
Press any key to continue . . .
```

setfill Manipulator

- Recall that in the manipulator **setw**, if the number of columns specified exceeds the number of columns required by the expression, the output of the expression is right-justified and the unused columns to the left are filled with spaces. The output stream variables can us the manipulator **setfill** to fill the unused columns with a character other than a space.
- The syntax to use the manipulator setfill is: **ostreamVar << setfill(ch);**
- Where **ostreamVar** is an output stream variable and **ch** is a character. For example, the statement: **cout << setfill('#');** sets the fill character to '#' on the standard output device.
- To use the manipulator **setfill**, the program must include the header file **iomanip**.

fixed.cpp | SetprecisionFixedShowpoint.cpp | setw.cpp | setfill.cpp

```cpp
//Example: setfill
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
    int x = 15; //Line 1
    int y = 7634; //Line 2
    cout << "12345678901234567890" << endl; //Line 3
    cout << setw(5) << x << setw(7) << y
         << setw(8) << "Warm" << endl; //Line 4
    cout << setfill('*'); //Line 5
    cout << setw(5) << x << setw(7) << y
         << setw(8) << "Warm" << endl; //Line 6
    cout << setw(5) << x << setw(7) << setfill('#')
         << y << setw(8) << "Warm" << endl; //Line 7
    cout << setw(5) << setfill('@') << x
         << setw(7) << setfill('#') << y
         << setw(8) << setfill('^') << "Warm"
         << endl; //Line 8
    cout << setfill(' '); //Line 9
    cout << setw(5) << x << setw(7) << y
         << setw(8) << "Warm" << endl; //Line 10
    return 0;
}
```

D:\Object Oriented Language\OOP Lab-07\OOP Lab-07 Programs\setfill.exe

```
12345678901234567890
   15   7634    Warm
***15***7634****Warm
***15###7634####Warm
@@@15###7634^^^^Warm
   15   7634    Warm

--------------------------------
Process exited after 0.08972 s
Press any key to continue . .
```

setfill Manipulator

- Recall that in the manipulator **setw**, if the number of columns specified exceeds the number of columns required by the expression, the output of the expression is **right-justified** and the unused columns to the left are filled with spaces. The output stream variables can us the manipulator **setfill** to fill the unused columns with a character other than a space.
- The syntax to use the manipulator setfill is: **ostreamVar << setfill(ch);**
- Where **ostreamVar** is an output stream variable and **ch** is a character. For example, the statement: **cout << setfill('#');** sets the fill character to '#' on the standard output device.
- To use the manipulator **setfill**, the program must include the header file **iomanip**.

fixed.cpp | setprecisionFixedShowpoint.cpp | setw.cpp | setfill.cpp

```
1  //Example: setfill
2  #include <iostream>
3  #include <iomanip>
4  using namespace std;
5  int main() {
6      int x = 15; //Line 1
7      int y = 7634; //Line 2
8      cout << "12345678901234567890" << endl; //Line 3
9      cout << setw(5) << x << setw(7) << y
10          << setw(8) << "Warm" << endl; //Line 4
11     cout << setfill('*'); //Line 5
12     cout << setw(5) << x << setw(7) << y
13          << setw(8) << "Warm" << endl; //Line 6
14     cout << setw(5) << x << setw(7) << setfill('#')
15          << y << setw(8) << "Warm" << endl; //Line 7
16     cout << setw(5) << setfill('@') << x
17          << setw(7) << setfill('#') << y
18          << setw(8) << setfill('^') << "Warm"
19          << endl; //Line 8
20     cout << setfill(' '); //Line 9
21     cout << setw(5) << x << setw(7) << y
22          << setw(8) << "Warm" << endl; //Line 10
23     return 0;
24 }
```

```
D:\Object Oriented Language\OOP Lab-07\OOP Lab-07 Programs\setfill.exe
12345678901234567890
   15   7634    Warm
***15***7634****Warm
***15###7634####Warm
@@@15###7634^^^^Warm
   15   7634    Warm

--------------------------------
Process exited after 0.08972 s
Press any key to continue . .
```

left and right Manipulator

- Recall that if the number of columns specified in the **setw** manipulator exceeds the number of columns required by the next expression the default output is **right-justified**. Sometimes, you might want the output to be **left-justified**. To left-justify the output, you use the manipulator **left**.
- The syntax to set the manipulator **left** is: **ostreamVar << left;** where **ostreamVar** is an output stream variable.
- For example, the following statement sets the output to be **left-justified** on the standard output device: **cout << left;**
- You can disable the manipulator **left** by using the stream function **unsetf**. The syntax to disable the manipulator **left** is:
- **ostreamVar.unsetf(ios::left);**
- The syntax to set the manipulator right is: **ostreamVar << right;** where **ostreamVar** is an output stream variable.
- For example, the following statement sets the output to be **right-justified** on the standard output device: **cout << right;**

Note:

- On some compliers, the statements
- **cin >> left; and cin >> right;** might not work.
- In this case, you can use
- **cin.setf(ios::left); in place of cin >> left;** and
- **cin.setf(ios::right); in place of cin >> right;.**

fixed.cpp | SetprecisionFixedShowpoint.cpp | setw.cpp | setfill.cpp | leftjustification.cpp

```
1  //Example: left justification
2  #include <iostream>
3  #include <iomanip>
4  using namespace std;
5  int main() {
6      int x = 15; //Line 1
7      int y = 7634; //Line 2
8      cout << left; //Line 3
9      cout << "12345678901234567890" << endl; //Line 4
10     cout << setw(5) << x << setw(7) << y
11          << setw(8) << "Warm" << endl; //Line 5
12     cout << setfill('*'); //Line 6
13     cout << setw(5) << x << setw(7) << y
14          << setw(8) << "Warm" << endl; //Line 7
15     cout << setw(5) << x << setw(7) << setfill('#')
16          << y << setw(8) << "Warm" << endl; //Line 8
17     cout << setw(5) << setfill('@') << x
18          << setw(7) << setfill('#') << y
19          << setw(8) << setfill('^') << "Warm"
20          << endl; //Line 9
21     cout << right; //Line 10
22     cout << setfill(' '); //Line 11
23     cout << setw(5) << x << setw(7) << y
24          << setw(8) << "Warm" << endl; //Line 12
25     return 0;
26 }
```

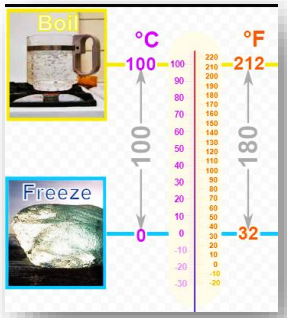
D:\Object Oriented Language\OOP Lab-07\OOP Lab-07 Programs\leftjustification.exe

```
12345678901234567890
15   7634   Warm
15***7634***Warm****
15***7634###Warm####
15@@@7634###Warm^^^^
   15   7634    Warm
```

DEBUGGING: "UNDERSTANDING LOGIC ERRORS AND DEBUGGING WITH COUT STATEMENTS"

Debugging: Understanding Logic Errors and Debugging with cout Statements

- As we have seen, syntax errors are reported by the compiler, and the compiler not only reports syntax errors, but also gives some explanation about the errors.
- On the other hand, logic errors are typically not caught by the compiler except for the trivial ones such as using a variable without properly initializing it.
- In this section, we illustrate how to spot and correct logic errors **using cout statements**.
- Suppose that we want to write a program that takes as input the temperature in **Fahrenheit** and outputs the equivalent temperature in **Celsius**. The formula to convert the temperature is: Celsius ¼ 5 / 9 * (Fahrenheit – 32). So consider the following program:



fixed.cpp | SetprecisionFixedShowpoint.cpp | setw.cpp | setfill.cpp | leftjustification.cpp | LogicError1.cpp

```
1  #include <iostream> //Line 1
2  using namespace std; //Line 2
3  int main() { //Line 3
4      //Line 4
5      int fahrenheit; //Line 5
6      int celsius; //Line 6
7      cout << "Enter temperature in Fahrenheit: "; //Lin
8      cin >> fahrenheit; //Line 8
9      cout << endl; //Line 9
10     celsius = 5 / 9 * (fahrenheit - 32); //Line 10
11     cout << fahrenheit << " degree F = "
12          << celsius << " degree C. " << endl; //Line 1
13     return 0; //Line 12
14 }
```

Select D:\Object Oriented Language\OOP Lab-07\OOP Lab-07 Programs\LogicError1.exe

```
Enter temperature in Fahrenheit: 32

32 degree F = 0 degree C.
```

D:\Object Oriented Language\OOP Lab-07\OOP Lab-07 Programs\LogicError1.exe

```
Enter temperature in Fahrenheit: 110

110 degree F = 0 degree C.
```

fixed.cpp | SetprecisionFixedShowpoint.cpp | setw.cpp | setfill.cpp | leftjustification.cpp | LogicError1.cpp | LogicError2.cpp

```
1  #include <iostream> //Line 1
2  using namespace std; //Line 2
3  int main() { //Line 3
4      //Line 4
5      int fahrenheit; //Line 5
6      int celsius; //Line 6
7      cout << "Enter temperature in Fahrenheit: "; //Line 7
8      cin >> fahrenheit; //Line 8
9      cout << endl; //Line 9
10     cout << "5 / 9 = " << 5 / 9
11          << "; fahrenheit - 32 = "
12          << fahrenheit - 32 << endl; //Line 9a
13     celsius = 5 / 9 * (fahrenheit - 32); //Line 10
14     cout << fahrenheit << " degree F = "
15          << celsius << " degree C. " << endl; //Line 11
16     return 0; //Line 12
17 } //Line 13
```

D:\Object Oriented Language\OOP Lab-07\OOP Lab-07 Programs\LogicError2.exe

```
Enter temperature in Fahrenheit: 110

5 / 9 = 0; fahrenheit - 32 = 78
110 degree F = 0 degree C.
```

fixed.cpp | SetprecisionFixedShowpoint.cpp | setw.cpp | setfill.cpp | leftjustification.cpp | LogicError1.cpp | LogicError2.cpp | LogicError3.cpp

```
1  #include <iostream> //Line 1
2  using namespace std; //Line 2
3  int main() { //Line 3
4      //Line 4
5      int fahrenheit; //Line 5
6      int celsius; //Line 6=
7      cout << "Enter temperature in Fahrenheit: "; //Line 7
8      cin >> fahrenheit; //Line 8
9      cout << endl; //Line 9
10     celsius = static_cast<int>
11          (5.0 / 9 * (fahrenheit - 32) + 0.5); //Line 10
12     cout << fahrenheit << " degree F = "
13          << celsius << " degree C. " << endl; //Line 11
14     return 0; //Line 12
15 } //Line 1
```

D:\Object Oriented Language\OOP Lab-07\OOP Lab-07 Programs\LogicError3.exe

```
Enter temperature in Fahrenheit: 110

110 degree F = 43 degree C.
```

FILE HANDLING
(FILE INPUT/OUTPUT)

File Input/output

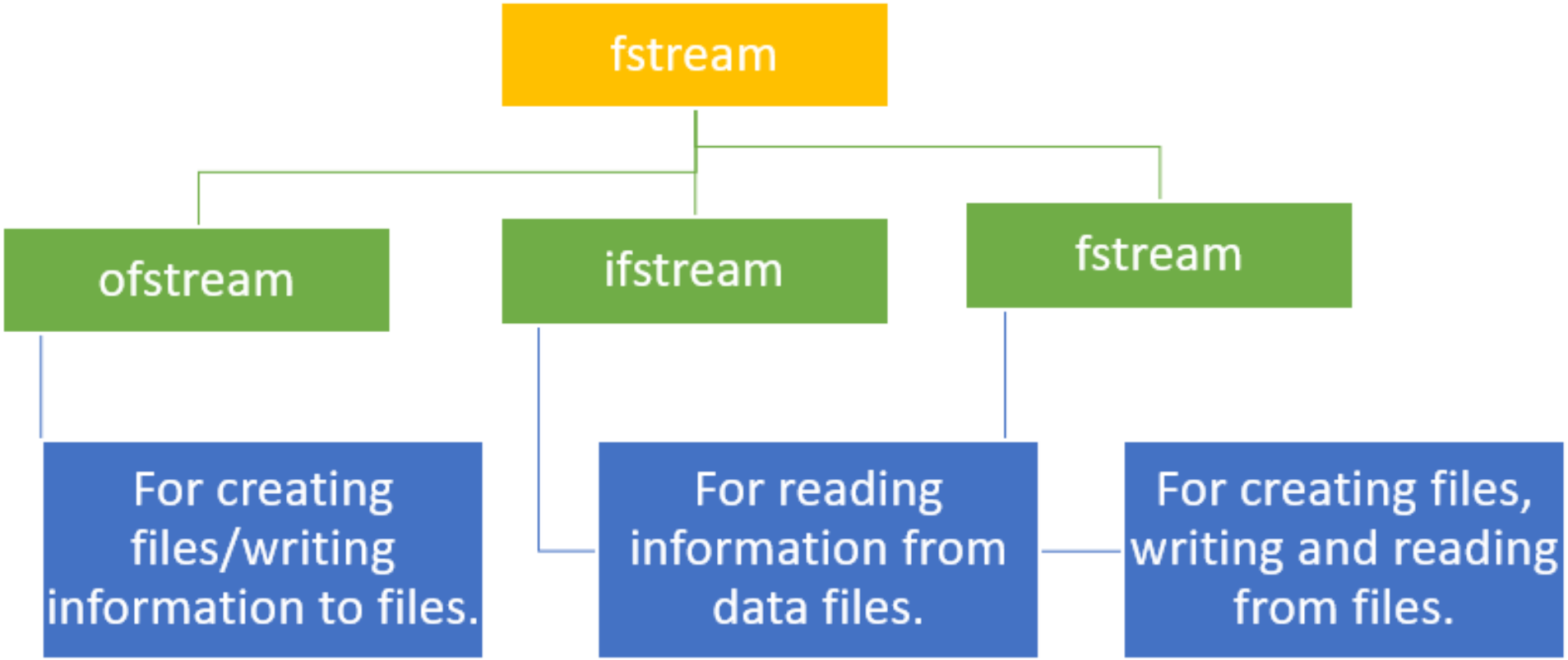
- **File:** An area in secondary storage used to hold information.
- C++ provides a header file called **fstream**, which is used for file I/O.
- Among other things, the **fstream** header file contains the definitions of two data types: **ifstream**, which means input file stream and is similar to **istream**, and **ofstream**, which means output file stream and is similar to ostream.\

- **What is file handling in C++?**
- Files store data permanently in a storage device. With file handling, the output from a program can be stored in a file. Various operations can be performed on the data while in the file.

- A stream is an abstraction of a device where input/output operations are performed. You can represent a stream as either a destination or a source of characters of indefinite length. This will be determined by their usage. C++ provides you with a library that comes with methods for file handling. Let us discuss it.
- **The fstream Library**
- The **fstream** library provides C++ programmers with three classes for working with files. These classes include:

- **ofstream**- This class represents an output stream. It's used for creating files and writing information to files.
- **ifstream**- This class represents an input stream. It's used for reading information from data files.
- **fstream**- This class generally represents a file stream. It comes with **ofstream/ifstream** capabilities. This means it's capable of creating files, writing to files, reading from data files.

File Input/output

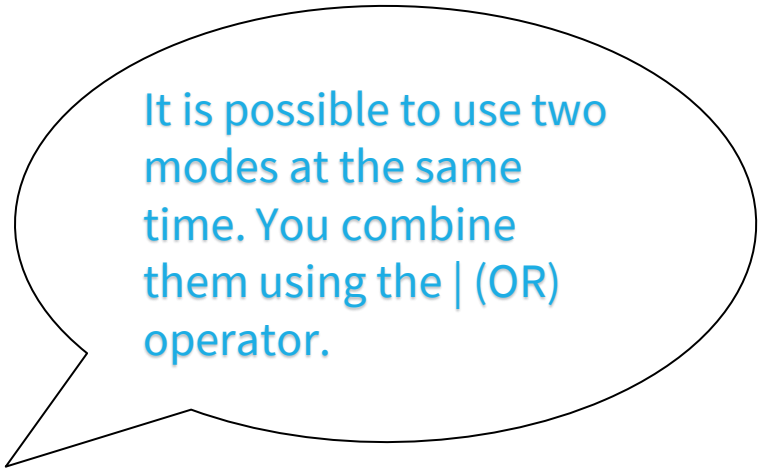


- To use the above classes of the **fstream** library, you must include it in your program as a header file. Of course, you will use the **#include preprocessor directive**. You must also include the **iostream** header file.

How to Open Files in C++

- Before performing any operation on a file, you must first open it. If you need to write to the file, open it using **fstream** or **ofstream** objects. If you only need to read from the file, open it using the **ifstream** object.
- The three objects, that is, **fstream**, **ofstream**, and **ifstream**, have the **open()** function defined in them. The function takes this syntax:
- **open (file_name, mode);**
- The **file_name** parameter denotes the name of the file to open e.g. "D:\Object Oriented Language\my_file.txt".
- The **mode** parameter is optional. It can take any of the following values:

Value	Description
ios:: app	The Append mode. The output sent to the file is appended to it.
ios::ate	It opens the file for the output then moves the read and write control to file's end.
ios::in	It opens the file for a read.
ios::out	It opens the file for a write.
ios::trunk	If a file exists, the file elements should be truncated prior to its opening.



How to Open(create) Files in C++

- To use the above classes of the **fstream** library, you must include it in your program as a header file. Of course, you will use the **#include preprocessor directive**. You must also include the **iostream** header file.

```
1  #include<iostream>
2
3  #include <fstream>
4
5  using namespace std;
6
7  int main() {
8
9      fstream my_file;
10
11     my_file.open("my_file", ios::out);
12
13     if (!my_file) {
14
15         cout << "File not created!";
16
17     }
18
19     else {
20
21         cout << "File created successfully!";
22
23         my_file.close();
24
25     }
26
27     return 0;
28
29 }
30
```

Code Explanation:

1. Include the iostream header file in the program to use its functions.
2. Include the **fstream** header file in the program to use its classes.
3. Include the std namespace in our code to use its classes without calling it.
4. Call the **main()** function. The program logic should go within its body.
5. Create an object of the **fstream** class and give it the name **my**_file.
6. Apply the **open()** function on the above object to create a new file. The out mode allows us to write into the file.
7. Use if statement to check whether file creation failed.
8. Message to print on the console if the file was not created.
9. End of the body of if statement.
10. Use an else statement to state what to do if the file was created.
11. Message to print on the console if the file was created.
12. Apply the **close()** function on the object to close the file.
13. End of the body of the else statement.
14. The program must return value if it completes successfully.
15. End of the **main()** function body.

How to Close Files in C++

- To use the above classes of the **fstream** library, you must include it in your program as a header file. Of course, you will use the **#include preprocessor directive**. You must also include the **iostream** header file.

Once a C++ program terminates, it automatically

- flushes the streams
- releases the allocated memory
- closes opened files.

However, as a programmer, you should learn to close open files before the program terminates.

The **fstream**, **ofstream**, and **ifstream** objects have the **close()** function for closing files. The function takes this syntax:

```
void close();
```

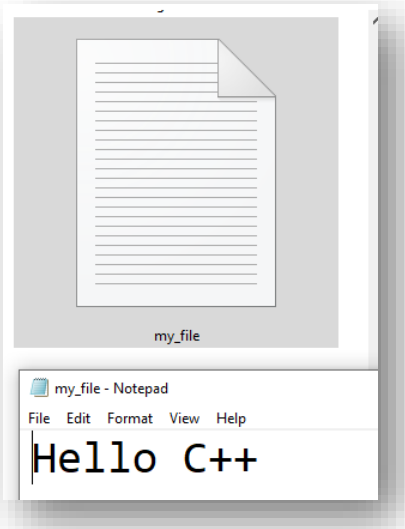
How to Write in the File in C++

- You can write to file right from your C++ program. You use stream insertion operator (<<) for this. The text to be written to the file should be enclosed within double-quotes.
- Let us demonstrate this.

```
1  #include <iostream>
2
3  #include <fstream>
4
5  using namespace std;
6
7  int main() {
8
9      fstream my_file;
10
11     my_file.open("my_file.txt", ios::out);
12
13     if (!my_file) {
14
15         cout << "File not created!";
16
17     }
18
19     else {
20
21         cout << "File created successfully!";
22
23         my_file << "Hello C++";
24
25         my_file.close();
26
27     }
28
29     return 0;
30
31 }
32
```

Code Explanation:

1. Include the iostream header file in the program to use its functions.
2. Include the **fstream** header file in the program to use its classes.
3. Include the std namespace in the program to use its classes without calling it.
4. Call the main() function. The program logic should be added within the body of this function.
5. Create an instance of the **fstream** class and give it the name **my_file.**
6. Use the **open()** function to create a new file named **my_file.txt**. The file will be opened in the out mode for writing into it.
7. Use an if statement to check whether the file has not been opened.
8. Text to print on the console if the file is not opened.
9. End of the body of the if statement.
10. Use an else statement to state what to do if the file was created.
11. Text to print on the console if the file was created.
12. Write some text to the created file.
13. Use the **close()** function to close the file.
14. End of the body of the else statement.
15. The program must return value upon successful completion.
16. End of the body of the **main()** function.



How to Read from Files in C++

- You can read information from files into your C++ program. This is possible using stream **extraction operator (>>)**. You use the operator in the same way you use it to read user input from the keyboard. However, instead of using the **cin object**, you use the **ifstream/ fstream** object.

```
1  #include <iostream>
2
3  #include <fstream>
4
5  using namespace std;
6
7  int main() {
8
9      fstream my_file;
10     my_file.open("my_file.txt", ios::in);
11
12     if (!my_file) {
13
14         cout << "No such file";
15     }
16     else {
17
18         char ch;
19
20         while (1) {
21             my_file >> ch;
22
23             if (my_file.eof())
24
25                 break;
26
27             cout << ch;
28         }
29
30     }
31     my_file.close();
32
33     return 0;
34 }
```

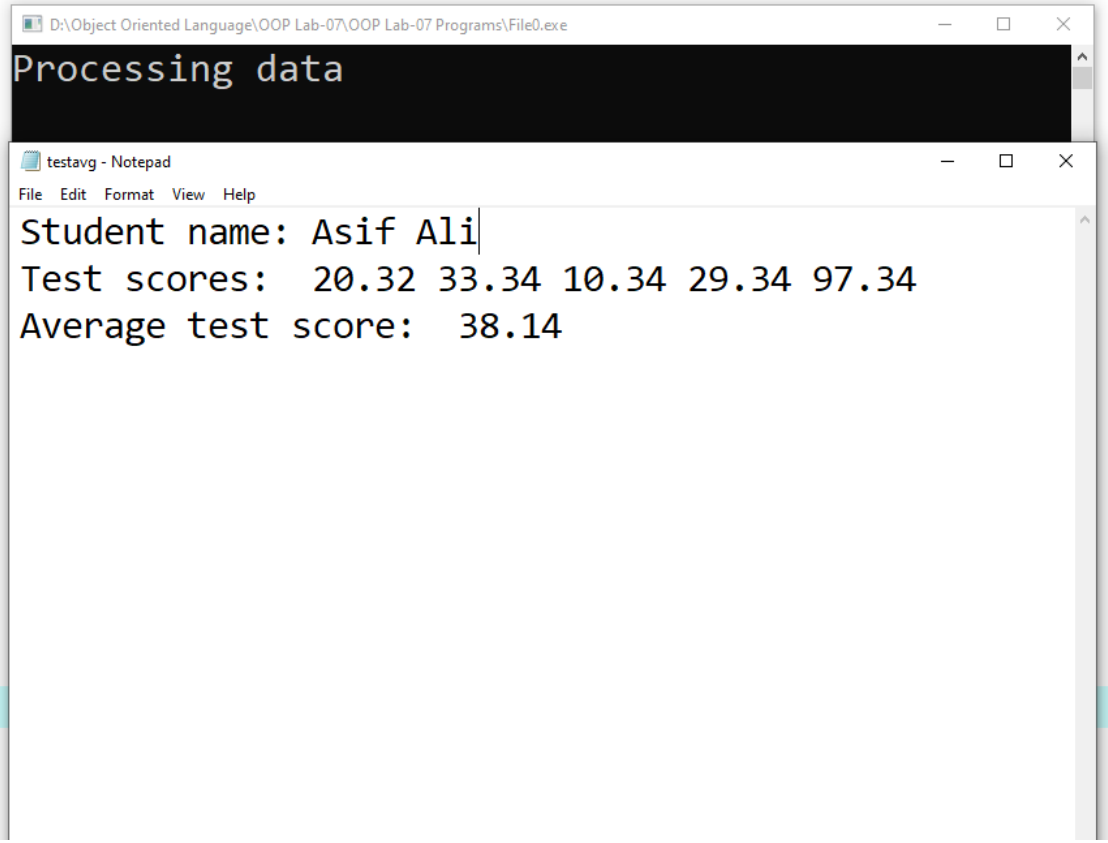
1.Include the iostream header file in the program to use its functions.
2.Include the **fstream** header file in the program to use its classes.
3.Include the std namespace in the program to use its classes without calling it.
4.Call the **main()** function. The program logic should be added within the body of this function.
5.Create an instance of the **fstream** class and give it the name **my_file**.
6.Use the **open()** function to create a new file named **my_file.txt**. The file will be opened in the in mode for reading from it.
7.Use an if statement to check whether the file does not exist.
8.Text to print on the console if the file is not found.
9.End of the body of the if statement.
10.Use an else statement to state what to do if the file is found.
11.Create a char variable named **ch**.
12.Create a while loop for iterating over the file contents.
13.Write/store contents of the file in the variable **ch**.
14.Use an if condition and **eof()** function that is, end of the file, to ensure the compiler keeps on reading from the file if the end is not reached.
15.Use a break statement to stop reading from the file once the end is reached.
16.Print the contents of variable **ch** on the console.
17.End of the while body.
18.End of the body of the else statement.
19.Call the **close()** function to close the file.
20.The program must return value upon successful completion.
21.End of the body of the **main()** function.

```
HelloC++
```

PROGRAMMING EXAMPLE: Student Grade

LogicError1.cpp | LogicError2.cpp | [*] LogicError3.cpp | File0.cpp | File1.cpp | File2.cpp | File3.cpp

```cpp
#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;
int main() {
    ofstream outFile;
    double test1 = 20.3234234, test2= 33.34234, test3= 10.34, test4= 29.3423423, test5= 97.34;
    double average= 20.34;
    string firstName= "Asif";
    string lastName= "Ali";
    outFile.open("testavg.out");
    outFile << fixed << showpoint;
    outFile << setprecision(2);
    cout << "Processing data" << endl;
    outFile << "Student name: " << firstName
            << " " << lastName << endl; //Step 6
    outFile << "Test scores: " << setw(6) << test1
            << setw(6) << test2 << setw(6) << test3
            << setw(6) << test4 << setw(6) << test5
            << endl; //Step 8
    average = (test1 + test2 + test3 + test4
               + test5) / 5.0; //Step 9
    outFile << "Average test score: " << setw(6)
            << average << endl; //Step 10
    outFile.close(); //Step 11
    return 0;
}
```



Summary File I/O

- **Summary:**
- With file handling, the output of a program can be sent and stored in a file.
- A number of operations can then be applied to the data while in the file.
- A stream is an abstraction that represents a device where input/output operations are performed.
- A stream can be represented as either destination or source of characters of indefinite length.
- The **fstream** library provides C++ programmers with methods for file handling.
- To use the library, you must include it in your program using the **#include preprocessor directive.**

Student Tasks Lab-07

1. Write a program in C++ to Find the Number of Lines in a Text File.
2. Write a program in C++: which will first store table into a file and then display on screen form the file.
   - **Program will take 3 input:**
     - **Table Value: 10**
     - **Final value: 8**
     - **File Name: Table of 10**

Text File

*Table of 10 - Notepad

File Edit Format View Help

```
Table of 10 is :
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
```

Output

```
Table of 10 is :
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
```

Student Tasks

3. Write C++ program determines the money to be donated to a charity.

- It prompts the user to input
  - The movie name,
  - Adult ticket price,
  - Child ticket price,
  - Number of adult tickets sold,
  - Number of child tickets sold, and
  - Percentage of the gross amount to be donated to the charity.
- Store output in a file (Use manipulator like setw, setfill, right, left, etc.)

```
grossAmount = adultTicketPrice * noOfAdultTicketsSold +
              childTicketPrice * noOfChildTicketsSold;
```

```
amountDonated = grossAmount * percentDonation / 100;
```

```
netSaleAmount = grossAmount - amountDonated;
```

- Output in the file must be look like

```
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
Movie Name: ....................... Journey to Mars
Number of Tickets Sold: ...........       2650
Gross Amount: ..................... $  9150.00
Percentage of Gross Amount Donated:      10.00%
Amount Donated: ................... $   915.00
Net Sale: ......................... $  8235.00
```

Reference

- **Chapter-03 of C++ programming: Form Problem Analysis to Program Design**
- **https://www.guru99.com/cpp-file-read-write-open.html**

THANKS ☺