- Programs need a way to output data on to a screen, file, or elsewhere. AN **ostream**, short for "output stream", is a class that supports output (via #include <\iostream>) and in the name space "std"
- ostream provides the **<< operator**, known as the **insertion operator** for converting different types of data into sequence of characters. That sequence is normally placed into a buffer, and the system then outputs the buffer at various times. The << operator returns a reference to the ostream that called the operator, and us evaluated from left to right like most operators, so << operators can appear in series
    - The << operator is overloaded w/ functions to support various standard data types (int, float, etc) each to a sequences of characters/
    - The operator can be further overloaded by the string library (#include string) or by the programmer for the programmer created classes.
- **cout** is a predefiend ostream object (declared of ostream cout; in the iostream library) that pre-assicuates with a system's standard ouput, usually on a computer screen.
    - Basic use of cout and insertion operator were covered earlier
- An **instream**, short for "input stream", is a class that supports input.
    - Avalible via #include <\iostream> to extract data from a data buffer and write data into different types of varibales
- **cin** is a predefined istream pre-associated with a system's standard input, usually a computer keyboard.
    - The system automatically puts the standard input into a data buffer associated with cin.
    - The >> operator skips leading whitespacce and exacts as many characters as possible consisting with the target variable's type.
    - Results are stored in a variable
- A **manipulator** is a function that overloads the insertion operator << or extraction operator >> to adjust the way output appears. Manipulators are defined in the ionmanip and ios libraries as namespace std.
    - Ex. fixed, scientific

Manipulator	Description	Example
fixed	Use fixed-point notation. From <iostream>	`// 12.340000` `cout << fixed << 12.34;`
scientific	Use scientific notation. From <iostream>	`// 1.234000e+01` `cout << scientific << 12.34;`

setprecision(p)	If stream has not been manipulated to fixed or scientific:
	Sets max number of digits in number
		`// 12.3`
		`cout << setprecision(3) << 12.34;`
		
		`// 12.34`
		`cout << setprecision(5) << 12.34;`

	If stream has been manipulated to fixed or scientific:
	Sets max number of digits in fraction only (after the decimal point).
	From <iomanip>
		`// 12.3`
		`cout << fixed << setprecision(1) << 12.34;`
		
		`// 1.2e+01`
		`cout << scientific << setprecision(1) << 12.34;`

showpoint	Even if fraction is 0, show decimal point and trailing 0s.
	Opposite is noshowpoint.
	From <iostream>
		`// 99`
		`cout << setprecision(3) << 99.0;`
		
		`// 99.0`
		`cout << setprecision(3) << showpoint << 99.0;`

- Text-alignment manipulators can also be used
  - They are persistent, once set it continue to next cout.

Manipulator	Description	Example

setw(n)	Sets the number of characters for the next output item only
	(does not persist, in contrast to other manipulators).
	By default, the item will be right-aligned, and filled with spaces.
	From <iomanip>
		`//  "    Amy"`
		`// " George"`
		`cout << setw(7) << "Amy" << endl;`
		`cout << setw(7) << "George" << endl;`

setfill(c)	Sets the fill to character c. From <iomanip>	`// "****Amy"` `cout << setfill('*') << setw(7) << "Amy";`
left	Changes to left alignment. From <iostream>	`// "Amy    "` `cout << left << setw(7) << "Amy";`
right	Changes back to right alignment. From <iostream>	`// "    Amy"` `cout << right << setw(7) << "Amy";`

- Buffer manipulators can also be used and executed through the programmer's choice of circumstance.

Manipulator	Description	Example
endl	Inserts a newline character '\n' into the output buffer and informs the system to flush the buffer. From <iostream>	`// Insert newline and flush` `cout << endl;`
flush	Informs the system to flush the buffer. From <iostream>	`// Flush buffer` `cout << flush;`

    - Using any of these, will immediately display text to output.
- A new **input string stream** variable of type **istringstream** can be created that reads input from an associated string instead of the keyboard (standard input). istringsteam is derived from istream. An istringstream can be used just like the cin stream as illustrated in the program below.
- Input streams have a function named **eof()** or end of file which return true or false depending on whether or not the stream has been reached
- An **output string stream** variable type of **ostringstream** can be insert characters into a string buffer instead of the screen. ostringstream is dervied from ostream. A program can insert characters into a ostringstream buffer using <<, just like the cout stream. The ostring member function **str()** returns the contents of an ostringstream buffer as a string
- **inFS.open(str)** is a function that has a string parameter str that specifies the name of the file to open. The file name can be a C__ string or a null-terminated C string. A progran can aslo be used as a user-enteredstring such as cin.
    - Using the **.fail()** fucntion can be used to check if the file was failed to be read.

Flag	Meaning	Function
goodbit	Indicates no error flags are set and the stream is good.	***good()*** returns true if no stream errors have occurred.
eofbit	Indicates if end-of-file reached on extraction.	***eof()*** returns value of eofbit, if end-of-file reached on extraction.
failbit	Indicates a logical error for the previous extraction or insertion operation.	***fail()*** returns true if either failbit or badbit is set, indicating an error for the previous stream operation.
badbit	Indicates an error occurred while reading or writing the stream, and the stream is bad. Further operations on the stream will fail.	***bad()*** returns true if badbit is set, indicating the stream is bad.

- An **ofstream**, short for "output file stream" is a clas that supports writing to a file. The ofstream class inherits from ostream.
- After declaring a variable of type ofstream, a file is opend using the ofstream function **open()** to check if the file oper successfully. Data can be writen using << and closed once finished using **close()**
-