CSE-4301

Object Oriented Programming

2022-2023

Week-10

Stream and Files

-Faisal Hussain

Department Of Computer Science And Engineering, IUT

Contents

Stream

- A *stream* is a general name given to a flow of data
- cin and cout are stream objects
- Different stream class represent different kinds of data flow

Advantage over C style file function:

- Simplicity. Use of format specifier is not necessary
- Overload insertion(<<) and extraction (>>) operators,
- they are the best way to write data to files, and also to format data in memory

standard stream

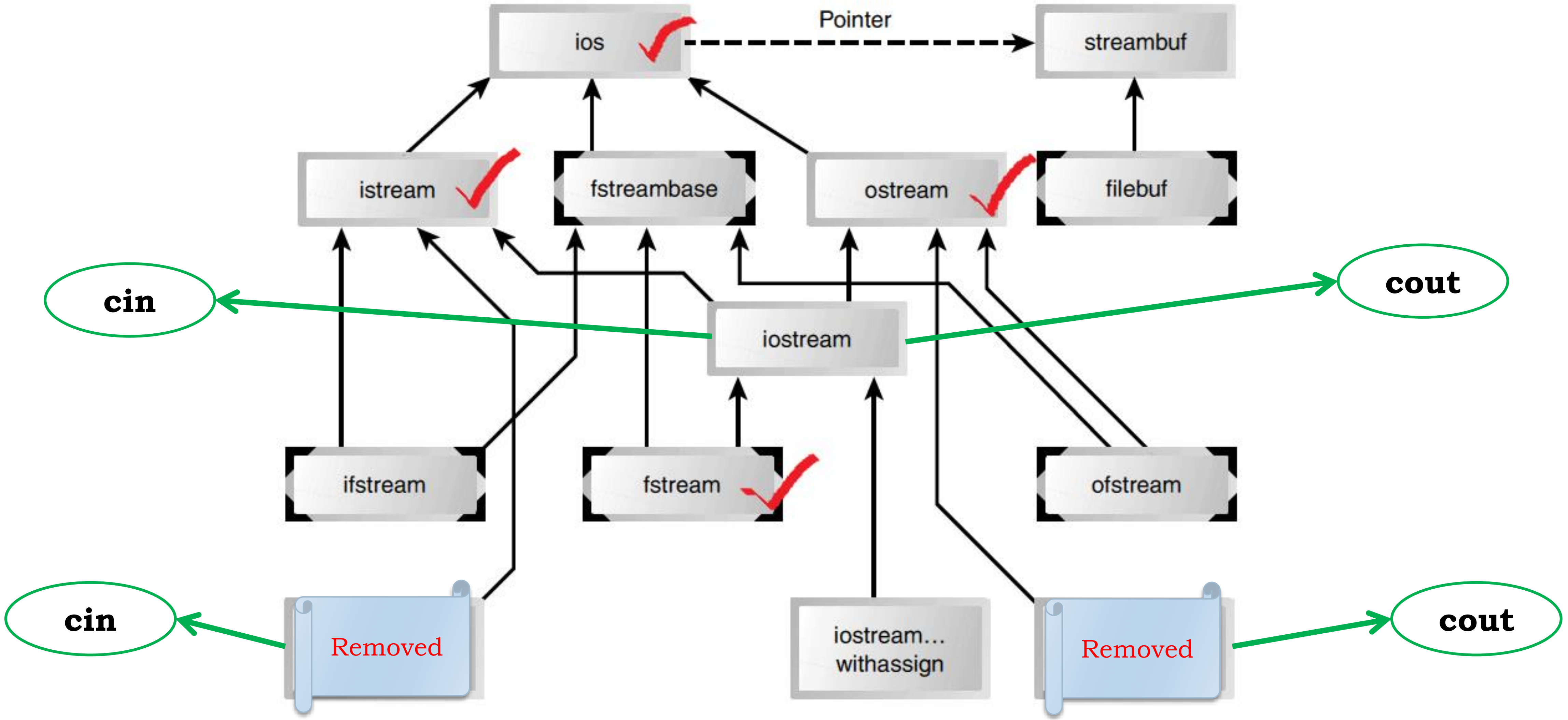
- **standard streams** are **pre-connected input and output channels between a computer program and its environment** (typically a text terminal) when it begins execution.
- In C : The three I/O connections are called
   1. standard input (**stdin**)
   2. standard output **(stdout**)
   3. standard error (**stderr).**

standard stream

- In C++ : The four I/O connections are called

1.	standard input **(cin)**	**Default device : Keyboard**
2.	standard output **(cout)**	**Default device : Screen**
3.	standard error **(cerr)**	**Default device : Screen**
4.	Buffered version of cerr **(clog).**	**Default device : Screen**

Stream Class Hierarchy



ios Class

- contains :
  1. **Formatting flags**
  2. **Error-status flags**
  3. **File operation mode**

Formatting flags

- A set of enum definitions in ios class.
- They act as on/off switches
- That specify choices for various aspects of input and output format and operation.

Formatting flags

field	member constant	effect when set
independent flags	`boolalpha`	read/write bool elements as alphabetic strings (`true` and `false`).
	`showbase`	write integral values preceded by their corresponding numeric base prefix.
	`showpoint`	write floating-point values including always the decimal point.
	`showpos`	write non-negative numerical values preceded by a plus sign (+).
	`skipws`	skip leading whitespaces on certain input operations.
	`unitbuf`	flush output after each inserting operation.
	`uppercase`	write uppercase letters replacing lowercase letters in certain insertion operations.
numerical base (`basefield`)	`dec`	read/write integral values using decimal base format.
	`hex`	read/write integral values using hexadecimal base format.
	`oct`	read/write integral values using octal base format.
float format (`floatfield`)	`fixed`	write floating point values in fixed-point notation.
	`scientific`	write floating-point values in scientific notation.
adjustment (`adjustfield`)	`internal`	the output is padded to the *field width* by inserting *fill characters* at a specified internal point.
	`left`	the output is padded to the *field width* appending *fill characters* at the end.
	`right`	the output is padded to the *field width* by inserting *fill characters* at the beginning.

Formatting flags

- **cout.setf (ios::left);** **// left justify output text**
- **cout << “This text is left-justified”;**
- **cout.unsetf (ios::left);** **// return to default (right justified)**

- Many formatting flags can be set using manipulators

Manipulators

- Manipulators are formatting instructions inserted directly into a stream.

- **endl:** which sends a newline to the stream and flushes it:
- **cout << “To each his own.” << endl<<“ ok”;**

- cout<< setiosflags(ios::fixed) // use fixed decimal point

          << setiosflags(ios::showpoint) // always show decimal point

          << var;

Manipulators

- Two Variant
  - No argument ios Manipulator
  - ios Manipulator with argument

Manipulator	*Purpose*
`ws`	Turn on whitespace skipping on input
`dec`	Convert to decimal
`oct`	Convert to octal
`hex`	Convert to hexadecimal
`endl`	Insert newline and flush the output stream
`ends`	Insert null character to terminate an output string
`flush`	Flush the output stream
`lock`	Lock file handle
`unlock`	Unlock file handle

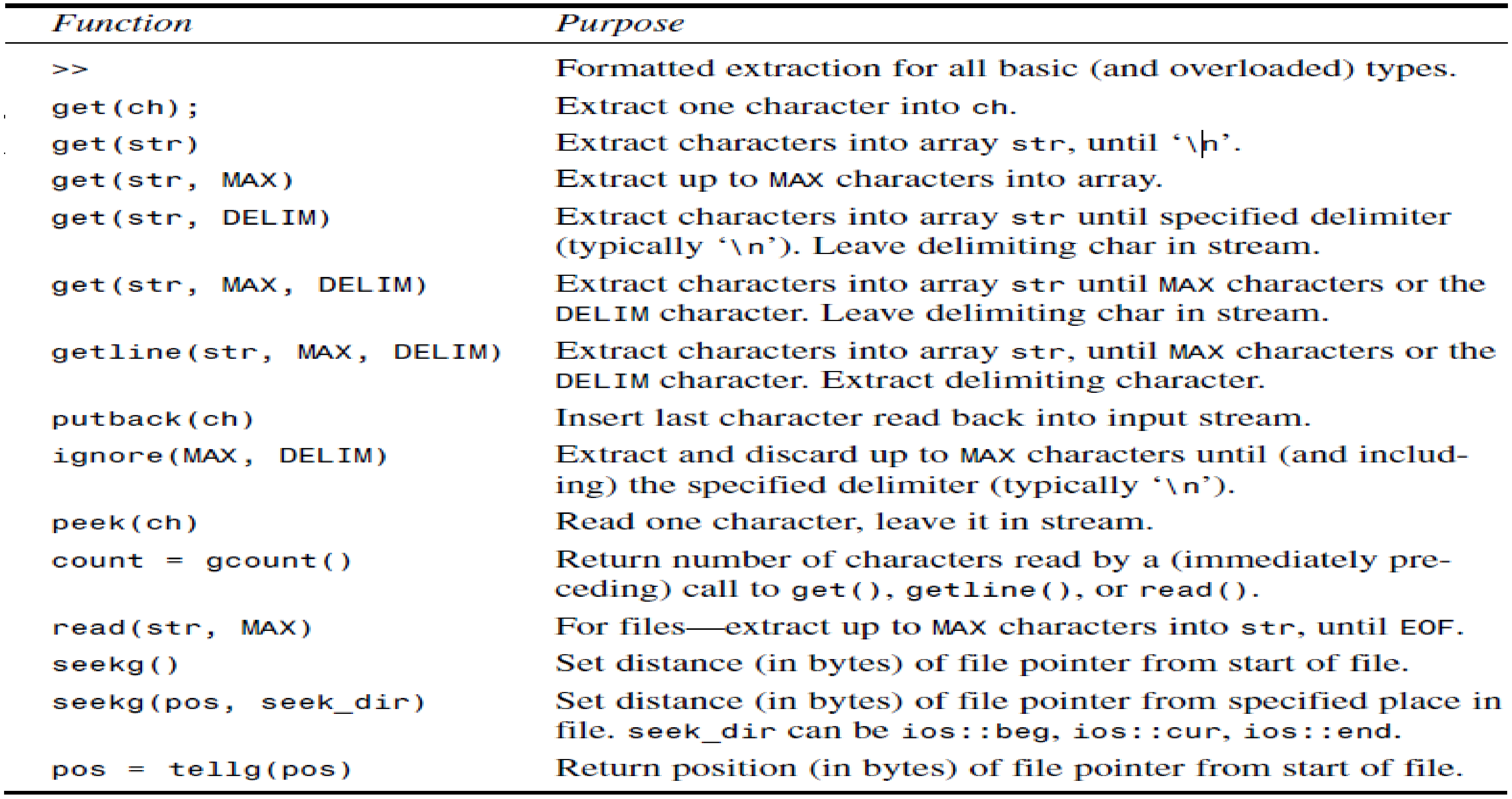
ios Manipulator with argument

Manipulator	*Argument*	*Purpose*
`setw()`	field width (`int`)	Set field width for output
`setfill()`	fill character (`int`)	Set fill character for output (default is a space)
`setprecision()`	precision (`int`)	Set precision (number of digits displayed)
`setiosflags()`	formatting flags (`long`)	Set specified flags
`resetiosflags()`	formatting flags (`long`)	Clear specified flags

ios Function

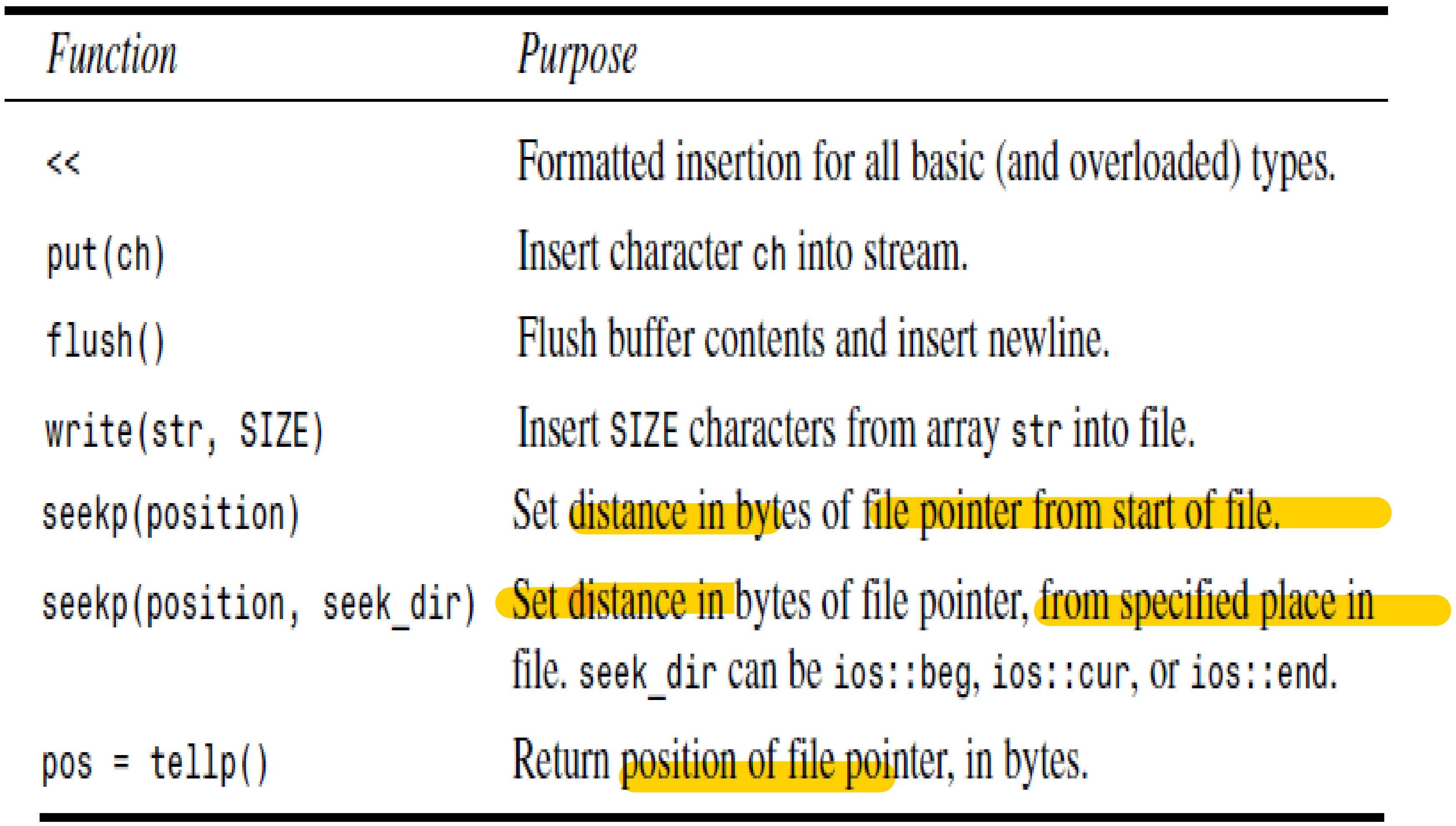
Function	*Purpose*
`ch = fill();`	Return the fill character (fills unused part of field; default is space)
`fill(ch);`	Set the fill character
`p = precision();`	Get the precision (number of digits displayed for floating-point)
`precision(p);`	Set the precision
`w = width();`	Get the current field width (in characters)
`width(w);`	Set the current field width
`setf(flags);`	Set specified formatting flags (for example, `ios::left`)
`unsetf(flags);`	Unset specified formatting flags
`setf(flags, field);`	First clear field, then set flags

istream function



Function	*Purpose*
`>>`	Formatted extraction for all basic (and overloaded) types.
`get(ch);`	Extract one character into `ch`.
`get(str)`	Extract characters into array `str`, until '`\n`'.
`get(str, MAX)`	Extract up to `MAX` characters into array.
`get(str, DELIM)`	Extract characters into array `str` until specified delimiter (typically '`\n`'). Leave delimiting char in stream.
`get(str, MAX, DELIM)`	Extract characters into array `str` until `MAX` characters or the `DELIM` character. Leave delimiting char in stream.
`getline(str, MAX, DELIM)`	Extract characters into array `str`, until `MAX` characters or the `DELIM` character. Extract delimiting character.
`putback(ch)`	Insert last character read back into input stream.
`ignore(MAX, DELIM)`	Extract and discard up to `MAX` characters until (and including) the specified delimiter (typically '`\n`').
`peek(ch)`	Read one character, leave it in stream.
`count = gcount()`	Return number of characters read by a (immediately preceding) call to `get()`, `getline()`, or `read()`.
`read(str, MAX)`	For files—extract up to `MAX` characters into `str`, until `EOF`.
`seekg()`	Set distance (in bytes) of file pointer from start of file.
`seekg(pos, seek_dir)`	Set distance (in bytes) of file pointer from specified place in file. `seek_dir` can be `ios::beg`, `ios::cur`, `ios::end`.
`pos = tellg(pos)`	Return position (in bytes) of file pointer from start of file.

ostream function



Function	*Purpose*
`<<`	Formatted insertion for all basic (and overloaded) types.
`put(ch)`	Insert character `ch` into stream.
`flush()`	Flush buffer contents and insert newline.
`write(str, SIZE)`	Insert `SIZE` characters from array `str` into file.
`seekp(position)`	Set distance in bytes of file pointer from start of file.
`seekp(position, seek_dir)`	Set distance in bytes of file pointer, from specified place in file. `seek_dir` can be `ios::beg`, `ios::cur`, or `ios::end`.
`pos = tellp()`	Return position of file pointer, in bytes.

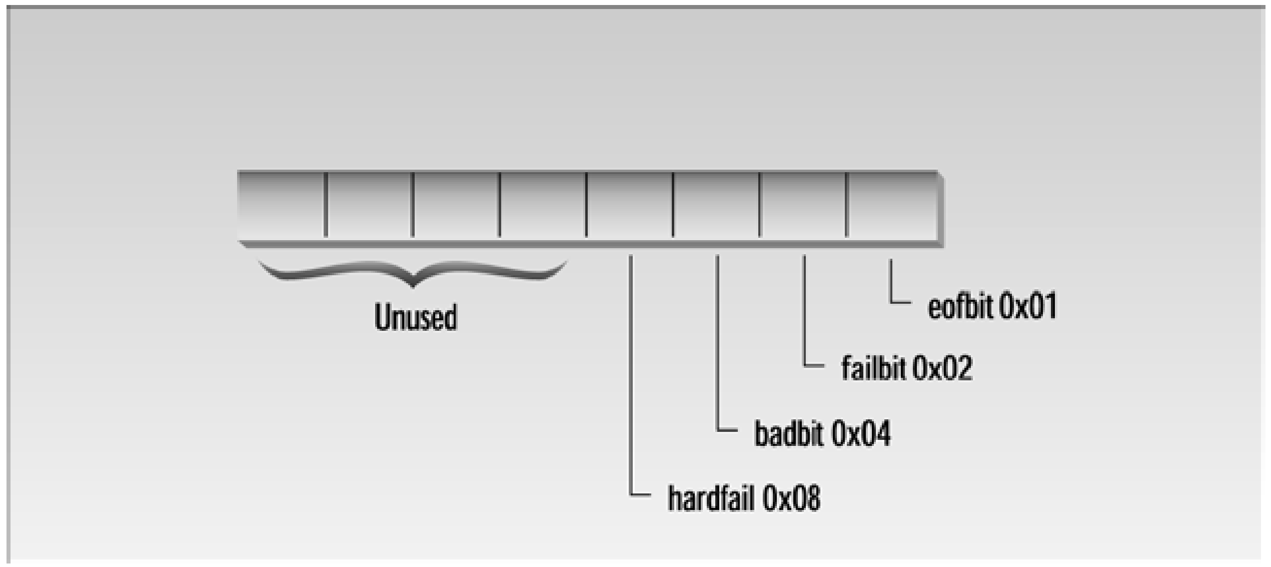
Error Status Bits

- The stream error-status flags constitute an ios enum member that reports errors that occurred in an input or output operation

Name	*Meaning*
`goodbit`	No errors (no flags set, value = 0)
`eofbit`	Reached end of file
`failbit`	Operation failed (user error, premature EOF)
`badbit`	Invalid operation (no associated streambuf)
`hardfail`	Unrecoverable error

Error Status Bits

Function	*Purpose*
`int = eof();`	Returns true if EOF flag set
`int = fail();`	Returns true if `failbit` or `badbit` or `hardfail` flag set
`int = bad();`	Returns true if `badbit` or `hardfail` flag set
`int = good();`	Returns true if everything OK; no flags set
`clear(int=0);`	With no argument, clears all error bits; otherwise sets specified flags, as in `clear(ios::failbit)`



Acknowledgement

- Ashraful Alam

  Assistant Professor,

  Department of Computer Science Engineering

  Islamic University of Technology