C++ File and Stream Input/Output

- File I/O is based on streams. <fstream.h> must be included in your program
- First operation is to open file

  ```
  ofstream outFile(char* filename,enum open_mode mode);
  ifstream inFile(char* filename,enum open_mode mode);
  ```

  open_mode is optional and defined in ios.h for appending, nocreate, noreplace
- Test if open operation was successful

  ```
  if(!outputFile)
      // error occurred during open
  ```
- Use << for output and >> for input
- Output and input operators defined for standard types
- Useful to provide << for your own classes

  ```
  friend ostream& opeator<<(ostream& s,YourClass& c)
  ```
- Use getline() or gets() members to read lines of text.

  ```
  inFile.getline(char* buffer,int size,char delimiter);
  inFile.gets(char** addrOfPtrToChar,char delimiter);
  ```

  With getline() you specify buffer and maximum size. With gets() you specify pointer to a pointer. gets() will provide a buffer with each call and store pointer in your pointer variable. Both functions will read newline from file but not place it in string.
- Test state of file operations using member functions

  ```
  while(!ifstreamObject.eof())
      // do something while reading data

  if(ofstreamObject.bad())
      // give error message that file output failed
  ```

  Also good() and fail() available. If stream is not good then all operations are null.
- There is a position associated with the file. This is the byte location in the file where the next read or write operation will take place.
- There are functions for manipulating the file position.

  ```
  // move to a specific byte position in file
  inFile.seekg(streampos);
  outFile.seekp(streampos);
  ```

```
// move relative to a direction.  seek_dir = beg, cur, end
// streamoff is offset from the seek_dir
inFile.seekg(streamoff,seek_dir);
outFile.seekp(streamoff,seek_dir);

// clear eof or bad state
inFile.clear();

// find file position
streampos inFile.tellg();
streampos outFile.tellg();
```

- You can look ahead in an input file or "put back" unwanted characters in the input buffers.

  ```
  int inFile.peek();
  inFile.putback(char c);
  ```

- There are two in-memory streams that are defined in <strstream.h>.

  ```
  char* p = new char(size);
  ostrstream outString(char *p, int size);
  istrstream inString(char *p,int size);
  ```

  All output is written to the string specified in the ostrstream create. Input is read from the istrstream. All stream operations are valid on these streams including stream position and state.

Output Formatting

- Many formatting commands are set via member functions
- Field width - minimum number of characters for next number or string

  ```
  cout.width(4);
  ```

- Fill character - specify the character to fill field. Default is blank.

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

- Many other format parameters can be set using setf() and unsetf() member functions.
  - Output base - ios::basefield {ios::oct, ios::dec, ios::hex}
  - Output justification - ios::adjustfield {ios::left,ios::right,ios::internal}
  - Floating point format - ios::floatfield {ios::scientific, ios::fixed}
  - Show plus sign - ios::showpos

  ```
  outFile.setf(ios::basefield,ios::hex);
  outFile.setf(ios::adjustfield,ios::right);
  outFile.unsetf(ios:showpos);
  ```