**ios::eof**

Return true if the eofbit error state flag is set for the stream.

This flag is set by all standard input operations when the End-of-File is reached in the sequence associated with the stream.

Note that the value returned by this function depends on the last operation performed on the stream ( and not on the next.)

Operations that attempt to read at the *End-Of-File* fail, and thus both the eofbit and the failbit end up set. This function can be used to check whether the failure is due to reaching the *End-Of-File* or to some other reason.

**ios::clear**

Sets a new value for the stream’s internal error state flags.

(stream의 flag를 재설정) -> stream buffer clear가 아니고 안에 parameter를 해당 stream의 goodbit, eofbit, failbit, badbit등 원하는 값으로 바꾸고 싶으면 clear.

The current value of the flags is overwritten : All bits are replaced by those in state; If state is goodbit(which is zero) all error flags are cleared.

In the case that no stream buffer is associated with the stream when this function is called, the badbit flag is automatically set(no matter the value for that bit passed in argument state).

Call하는 stream이 없는 경우에 그 stream instance로 call한 경우-> 자동으로 badbit flag가 set 된다.

Note that changing the state may throw an exception, depending on the latest settings passed to member exceptions.

The current state can be obtained with member function rdstate.

-> 해당 stream object의 function rdstate(abbreviation : read state)로 현재 flag의 state를 확인할 수 있음

<File Input Stream 생성 in C++>

--------------------------------------------------------------------------

ifstream으로 파일 입력 코드 짤 때, 순서가

std::ifstream var\_name(“file path+name”); // 으로 ifstream object 생성

예외처리 var\_name.is\_open()으로 file existence 확인

!var\_name.eof()로 while loop 돌면서 cin 처럼 var\_name으로 >> 활용

끝마무리는 var\_name.close();

--------------------------------------------------------------------------

#include <algorithm> std::sort class 존재

template <class RandomAcessIterator>

void sort(RandomAcessIterator first, RandomAccessIterator last);

template <class RandomAccessIterator, class Compare>

void sort(RandomAccessIterator first, RandomAccessIterator last, Compare comp);

comp

-> Binary function that accepts two elements in the range as arguments, and returns a value convertible to bool. The value returned indicates whether the element passed as first argument is considered to go before the second in the specific strict weak ordering it defines. The function shall not modify any of its arguments. This can either be a function pointer or a function object.

strict weak ordering

🡪 Two objects x and y are equivalent if both f(x,y) and f(y,x) are false. Note that an object is always (by the irreflexivity invariant) requivalent to itself. irreflexivity : 반사적인지 않은?

link : [https://en.wikipedia.org/wiki/Weak\_ordering#Strict\_weak\_orderings](https://en.wikipedia.org/wiki/Weak_ordering%23Strict_weak_orderings)

1)a==b

a<b -> false

b<a -> false

2)a<b

a<b -> true

b<a -> false

3)b<a

a<b -> false

b<a -> truue

Sort elements in range

Sorts the elemeents in the range [first,last] into asceding order.

The elements are compared using operator< for the first version, and comp for the second.

Equivalent elements are not guaranteed to keep their original relative order(1,4,2,2,3)

0,1,2,3,4

2와 2간의 relative-order가 여기서는 iterator[2]와 iterator[3]으로 유지를 보장못한다는 의미.

-> 이거도 유지하고 싶으면 stable\_sort