



고급프로그래밍

파일 입출력

Professor Jeong, Mun-Ho

Robot Vision & Intelligence Laboratory
Kwangwoon University
(02-940-5625, mhjeong@kw.ac.kr)

Schedule

주차	주제		과제	퀴즈
1	과목소개	교과목 소개 (1), C++ 시작 (2)		
2	C++	C++ 프로그래밍의 기본 (3), 클래스와 객체 (4)	1	1
3		객체생성과 사용 (5)	2	2
4		함수와 참조 (6, 3/26), 복사 생성자와 함수중복(7)	3	3
5		static, friend, 연산자중복 (8, 4/2), 연산자중복 상속(9)	4	
6		상속 (10, 4/9), 가상함수와 추상클래스 (11)		4
7		템플릿과 STL (12, 4/16), 입출력(13)	5	
8		중간고사		
9		파일 입출력(14), 예외처리 및 C 사용, 람다식		
10		멀티스레딩		
11		멀티스레딩, 고급문법		
12		고급문법		
13	병렬프로그래밍	병렬프로그래밍		
14		병렬프로그래밍		
15		기말고사		

오늘의 학습내용

- 파일 입출력

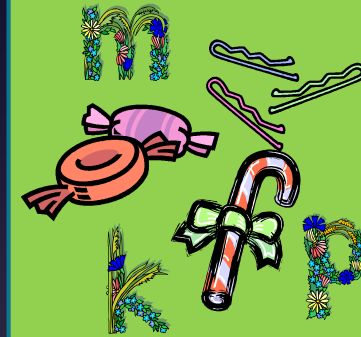
파일 입출력

텍스트 파일과 바이너리 파일

길을 걷고 산들 무엇 하나.
꽃이 내가 아니듯
내가 꽃이 될 수 없는 지금
물빛 몸매를 가진
한 마리 학으로
살아 무엇 하나.

문자만으로 구성된
문서

텍스트 파일



바이너리 파일

문자, 그림, 표, 사운드, 동영상
등으로 구성된 문서

텍스트 파일

■ 텍스트 파일

- 사람들이 사용하는 글자 혹은 문자들로만 구성되는 파일
 - 알파벳, 한글, 숫자, % # @ < ? 등의 기호 문자
- '\n', '\t' 등의 특수 문자도 포함
- 각 문자마다 문자 코드(이진수) 할당
 - ASCII 코드, 유니코드
- 텍스트 파일의 종류
 - txt 파일, HTML 파일, XML 파일, C++ 소스 파일, C 소스 파일, 자바 소스 파일

■ 텍스트 파일과 <Enter> 키

- <Enter>키를 입력하면 텍스트 파일에는 '\r', '\n'의 두 코드가 기록됨

ASCII 코드 표 샘플

텍스트 파일

10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	29	(72	48	H	104	68	h
9	09	Horizontal tab	41	29)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out				78	4E	N	110	6E	n

텍스트 파일의 내부

<Enter> 키

'Wr' 'Wn'

```
elvis - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
Wise men say,
only fools rush in
But I can't help,
falling in love with you

Shall I stay?
Would it be a sin?
If I can't help,
falling in love with you

Like a river flows,
surely to the sea
Darling so it goes,
some things aren't meant to be

Take my hand,
take my whole life too
For I can't help,
falling in love with you - 1961
```

00000000	57	69	73	65	20	6D	65	6E	20	73	61	79	2C	0D	0A	6F	Wise men say,..o
00000010	6E	6C	79	20	66	6F	6F	6C	73	20	72	75	73	68	20	69	nly fools rush i
00000020	6E	0D	0A	42	75	74	20	49	20	63	61	6E	27	74	20	68	n..But I can't h
00000030	65	6C	70	2C	0D	0A	66	61	6C	6C	69	6E	67	20	69	6E	elp,..falling in
00000040	20	6C	6F	76	65	20	77	69	74	68	20	79	6F	75	0D	0A	love with you..
00000050	0D	0A	53	68	61	6C	6C	20	49	20	73	74	61	79	3F	0D	..Shall I stay?.
00000060	0A	57	6F	75	6C	64	20	69	74	20	62	65	20	61	20	73	.Would it be a s
00000070	69	6E	3F	0D	0A	49	66	20	49	20	63	61	6E	27	74	20	in?..If I can't
00000080	68	65	6C	70	2C	0D	0A	66	61	6C	6C	69	6E	67	20	69	help,..falling i
00000090	6E	20	6C	6F	76	65	20	77	69	74	68	20	79	6F	75	0D	n love with you.
000000a0	0A	0D	0A	4C	69	6B	65	20	61	20	72	69	76	65	72	20	...Like a river
000000b0	66	6C	6F	77	73	2C	0D	0A	73	75	72	65	6C	79	20	74	flows,..surely t
000000c0	6F	20	74	68	65	20	73	65	61	0D	0A	44	61	72	6C	69	o the sea..Darli
000000d0	6E	67	20	73	6F	20	69	74	20	67	6F	65	73	2C	0D	0A	ng so it goes,..
000000e0	73	6F	6D	65	20	74	68	69	6E	67	73	20	61	72	65	6E	some things aren
000000f0	27	74	20	6D	65	61	6E	74	20	74	6F	20	62	65	0D	0A	't meant to be..
00000100	0D	0A	54	61	6B	65	20	6D	79	20	68	61	6E	64	2C	0D	..Take my hand,..
00000110	0A	74	61	6B	65	20	6D	79	20	77	68	6F	6C	65	20	6C	.take my whole l
00000120	69	66	65	20	74	6F	6F	0D	0A	46	6F	72	20	49	20	63	ife too..For I c
00000130	61	6E	27	74	20	68	65	6C	70	2C	0D	0A	66	61	6C	6C	an't help,..fall
00000140	69	6E	67	20	69	6E	20	6C	6F	76	65	20	77	69	74	68	ing in love with
00000150	20	79	6F	75	20	2D	20	31	39	36	31	0D	0A				you - 1961..

파일 내의 주소

파일 내부의 바이너리 데이터
(16진수로 표현됨)

문자로 표현

elvis.txt를 Edit Plus로 열어 놓은 화면

자세히 보기

<Enter> 키 한 번에 두 개의 제어 문자 삽입

0x0D -> 'Wr'

0x0A -> 'Wn'

0x20 -> 스페이스 '

00000000	57	69	73	65	20	6D	65	6E	20	73	61	79	2C	0D	0A	6F	Wise men say,..o
00000010	6E	6C	79	20	66	6F	6F	6C	73	20	72	75	73	68	20	69	nly fools rush i
00000020	6E	0D	0A	42	75	74	20	49	20	63	61	6E	27	74	20	68	n..But I can't h
00000030	65	6C	70	2C	0D	0A	66	61	6C	6C	69	6E	67	20	69	6E	elp,..falling in
00000040	20	6C	6F	76	65	20	77	69	74	68	20	79	6F	75	0D	0A	love with you..
00000050	0D	0A	53	68	61	6C	6C	20	49	20	73	74	61	79	3F	0D	..Shall I stay?.
00000060	0A	57	6F	75	6C	64	20	69	74	20	62	65	20	61	20	73	.Would it be a s
00000070	69	6E	3F	0D	0A	49	66	20	49	20	63	61	6E	27	74	20	in?..If I can't
00000080	68	65	6C	70	2C	0D	0A	66	61	6C	6C	69	6E	67	20	69	help,..falling i
00000090	6E	20	6C	6F	76	65	20	77	69	74	68	20	79	6F	75	0D	n love with you.
000000a0	0A	0D	0A	4C	69	6B	65	20	61	20	72	69	76	65	72	20	...Like a river
000000b0	66	6C	6F	77	73	2C	0D	0A	73	75	72	65	6C	79	20	74	flows,..surely t
000000c0	6F	20	74	68	65	20	73	65	61	0D	0A	44	61	72	6C	69	o the sea..Darli
000000d0	6E	67	20	73	6F	20	69	74	20	67	6F	65	73	2C	0D	0A	ng so it goes,..
000000e0	73	6F	6D	65	20	74	68	69	6E	67	73	20	61	72	65	6E	some things aren
000000f0	27	74	20	6D	65	61	6E	74	20	74	6F	20	62	65	0D	0A	't meant to be..
00000100	0D	0A	54	61	6B	65	20	6D	79	20	68	61	6E	64	2C	0D	..Take my hand,.
00000110	0A	74	61	6B	65	20	6D	79	20	77	68	6F	6C	65	20	6C	.take my whole l
00000120	69	66	65	20	74	6F	6F	0D	0A	46	6F	72	20	49	20	63	ife too..For I c
00000130	61	6E	27	74	20	68	65	6C	70	2C	0D	0A	66	61	6C	6C	an't help,..fall
00000140	69	6E	67	20	69	6E	20	6C	6F	76	65	20	77	69	74	68	ing in love with
00000150	20	79	6F	75	20	2D	20	31	39	36	31	0D	0A				you - 1961..

0x31 -> 문자 '1'

바이너리 파일

■ 바이너리 파일이란?

- 문자로 표현되지 않는 바이너리 데이터가 기록된 파일
 - 이미지, 오디오, 그래픽, 컴파일된 코드는 문자로 표현되지 않음
- 텍스트 파일의 각 바이트 -> 문자로 해석
- 바이너리 파일의 각 바이트 -> 문자로 해석되지 않는 것도 있음
 - 각 바이트의 의미는 파일을 만든 응용프로그램 만이 해석 가능

■ 바이너리 파일의 종류

- jpeg, bmp 등의 이미지 파일
- mp3 등의 오디오 파일
- hwp, doc, ppt 등의 확장자를 가진 멀티미디어 문서 파일
- obj, exe 등의 확장자를 가진 컴파일된 코드나 실행 파일

바이너리 파일의 내부



```

00000000 FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 01 .....JFIF.....
00000010 00 01 00 00 FF DB 00 84 00 09 06 06 14 12 11 15 .....
00000020 14 13 14 16 15 15 16 17 1A 17 17 18 17 18 1D 1B .....
00000030 1A 18 1D 19 1C 18 17 1E 1F 1F 1C 18 1D 26 1E 18 .....&..
00000040 1C 25 1A 17 18 1F 2F 20 23 27 29 2C 2C 2C 18 1E .$. / #' ), , , , .
00000050 31 35 30 2A 35 26 2B 2C 29 01 09 0A 0A 0E 0C 0E 150*5&+, , , , .
00000060 1A 0F 0F 1A 2C 1C 1C 24 2C 29 29 29 2A 29 29 2C .....$
00000070 29 29 29 2C 29 2D 29 29 29 2C 2C 29 29 29 29 29 ))) , ) - )
00000080 2C 2C 2C 2C 29 29 2C 2C 29 2C 29 29 29 2C 29 2C , , , , ) , , ,
00000090 29 2C 2C 29 29 31 2C 29 2C 29 FF C0 00 11 08 00 ), , , ) 1 , ) , , , .
000000a0 BA 01 0F 03 01 22 00 02 11 01 03 11 01 FF C4 00 .....".
000000b0 1C 00 00 02 03 01 01 01 01 00 00 00 00 00 00 00 .....
000000c0 00 00 04 05 02 03 06 07 01 00 08 FF C4 00 4A 10 .....J.
000000d0 00 02 01 02 04 03 06 02 06 07 06 04 03 09 00 00 .....
000000e0 01 02 11 00 03 04 12 21 31 05 41 51 06 13 22 61 .....!1.AQ.."a
000000f0 71 81 32 91 42 52 A1 B1 C1 D1 07 14 23 62 72 E1 q.2.BR.....#br.
00000100 F0 43 82 92 A2 B2 F1 15 24 33 53 34 74 C2 25 54 .C.....$3S4t.$T
00000110 63 73 83 93 A3 A4 E2 FF C4 00 1A 01 00 02 03 01 cs.....
00000120 01 00 00 00 00 00 00 00 00 00 00 00 02 03 01 04 .....
00000130 05 00 06 FF C4 00 2A 11 00 02 02 02 02 02 00 06 .....*.
00000140 02 02 03 00 00 00 00 00 00 01 02 11 03 21 12 31 .....!.1
00000150 04 41 05 13 22 32 51 61 71 81 14 91 23 B1 F0 FF .A.."2Qaq...#...

```

문자로 매핑되지
않는 바이너리 값

파일 내의 주소

파일 내부의 바이너리 데이터
(16진수로 표현됨)

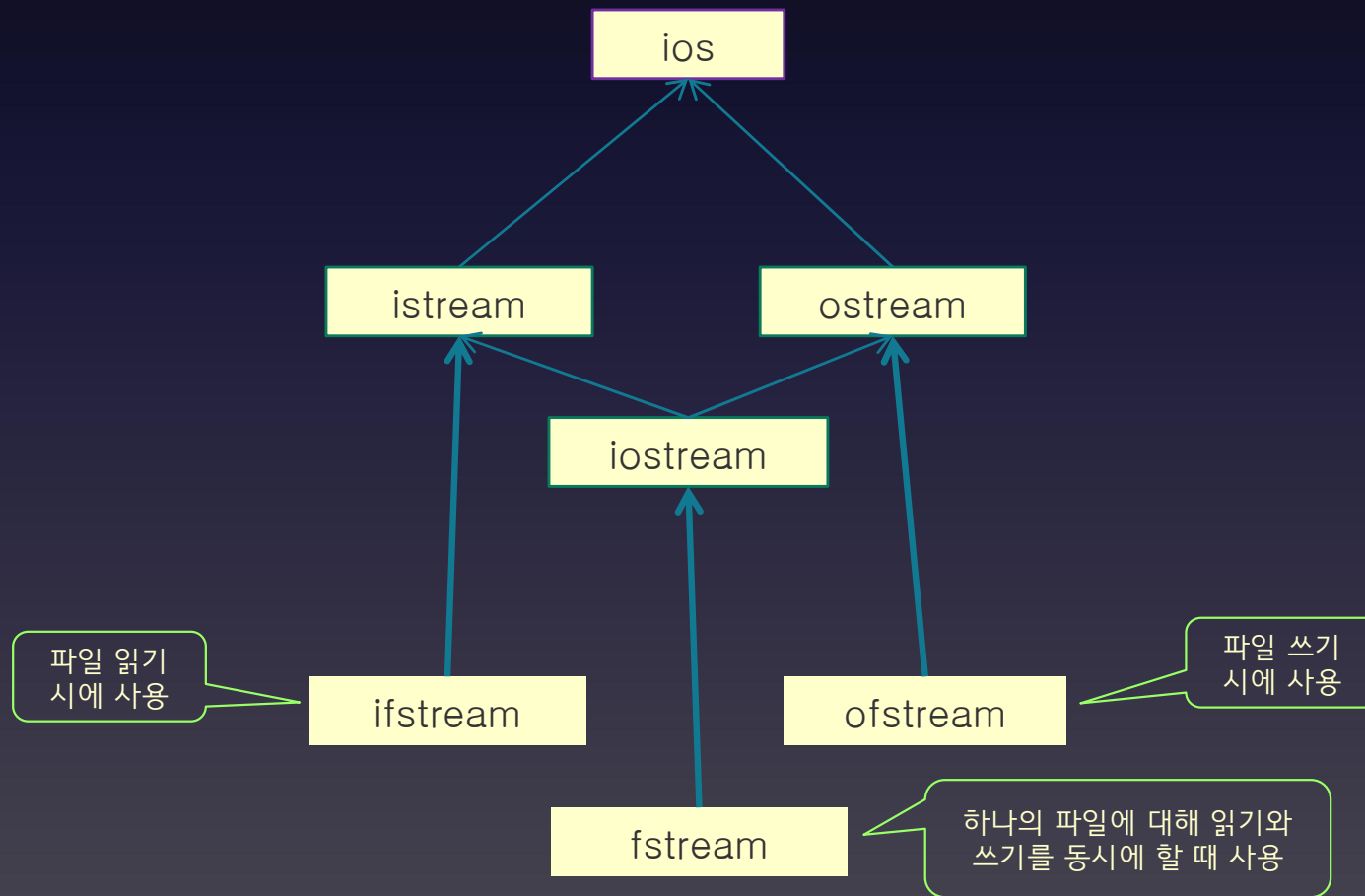
uisee.jpg를 Edit Plus로 열어 보는 화면

hwp 파일 텍스트 파일인가

- hwp 파일은 바이너리 파일
 - 텍스트 정보 포함
 - 한글이나 영어 문자 포함
 - 바이너리 정보 포함
 - 글자 색이나 서체 등의 문자 포맷 정보
 - 비트맵 이미지
 - 표
 - 선, 원 등의 그래픽 정보
 - 왼쪽 마진, 오른쪽 마진 등 문서 포맷 정보

C++ 표준 파일 입출력 라이브러리

- 스트림 입출력 방식 지원



파일 입출력 스트림

- >> 연산자와 istream의 get, read() 함수
 - 연결된 장치로부터 읽는 함수
 - 키보드에 연결되면 키 입력을, 파일에 연결되면 파일에서 입력
- << 연산자와 ostream의 put(), write() 함수
 - 연결된 장치에 쓰는 함수
 - 스크린에 연결되면 화면에, 파일에 연결되면 파일에 출력



헤더 파일과 namespace

- C++ 파일 입출력 라이브러리 사용
 - <fstream> 헤더 파일과 std 이름 공간의 선언 필요

```
#include <fstream>  
using namespace std;
```

파일 입출력 모드

■ 파일 입출력 방식

- 텍스트 I/O와 바이너리 I/O의 두 방식
 - C++ 파일 입출력 클래스(ifstream, ofstream, fstream)는 두 방식 지원

■ 텍스트 I/O

- 문자 단위로 파일에 쓰기, 파일에서 읽기
 - 문자를 기록하고, 읽은 바이트를 문자로 해석
- 텍스트 파일에만 적용

■ 바이너리 I/O

- 바이트 단위로 파일에 쓰기, 파일에서 읽기
 - 데이터를 문자로 해석하지 않고 있는 그대로 기록하거나 읽음
- 텍스트 파일과 바이너리 파일 모두 입출력 가능

■ 텍스트 I/O와 바이너리 I/O 입출력 시 차이점

- 개행 문자('\n')를 다루는데 있음(뒤에서 설명)

<< 연산자를 이용한 간단한 파일 출력

파일 쓰기 위한 스트림 생성

ofstream fout;

파일 열기

fout.open("song.txt"); // song.txt 파일 열기

ofstream fout("song.txt");
한 줄로 줄여 쓸 수 있음

파일 열기 성공 검사.
operator !() 실행

if(!fout) { // fout 스트림의 파일 열기가 실패한 경우
// 파일 열기 실패를 처리하는 코드
}

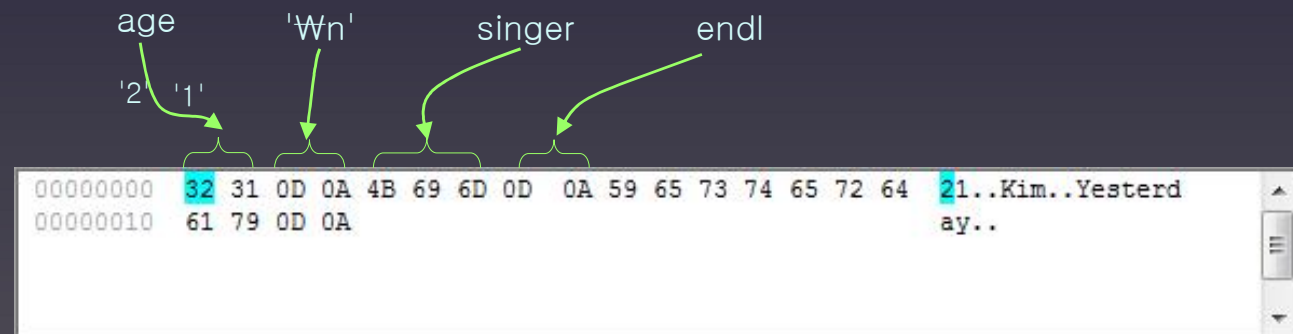
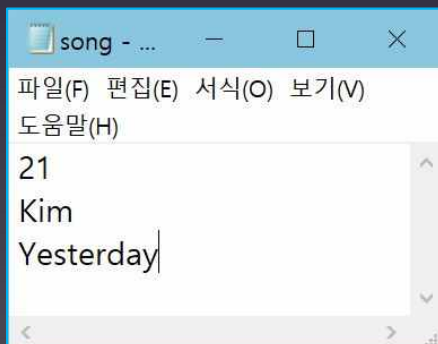
if(!fout.is_open())과 동일

파일 쓰기

int age = 21;
char singer[] = "Kim";
char song[] = "Yesterday";
fout << age << 'Wn'; // 파일에 21과 'Wn'을 기록한다.
fout << singer << endl; // 파일에 "Kim"과 'Wn'을 덧붙여 기록한다.
fout << song << endl; // 파일에 "Yesterday"와 'Wn'을 덧붙여 기록한다.

파일 닫기

fout.close();



예제

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    char name[10], dept[20];
    int sid;

    // 키보드로부터 읽기
    cout << "이름>>";
    cin >> name; // 키보드에서 이름 입력 받음
    cout << "학번>>";
    cin >> sid; // 키보드에서 학번 입력 받음
    cout << "학과>>";
    cin >> dept; // 키보드에서 학과 입력 받음
```

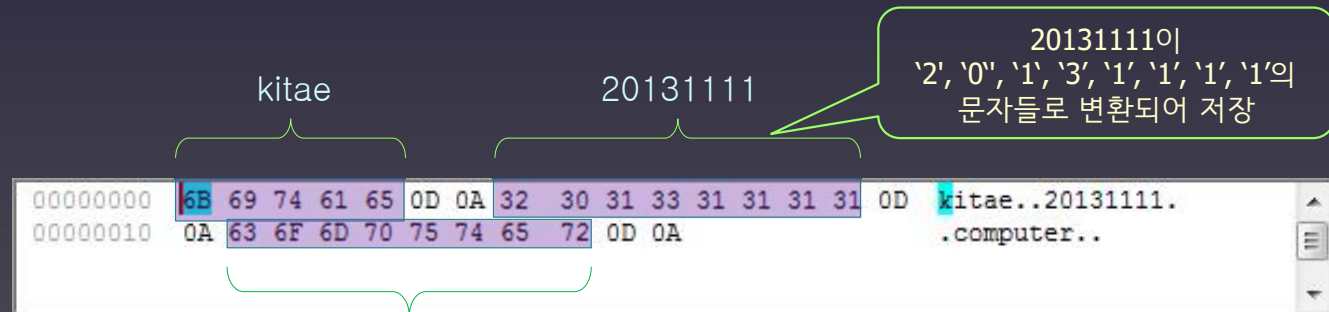
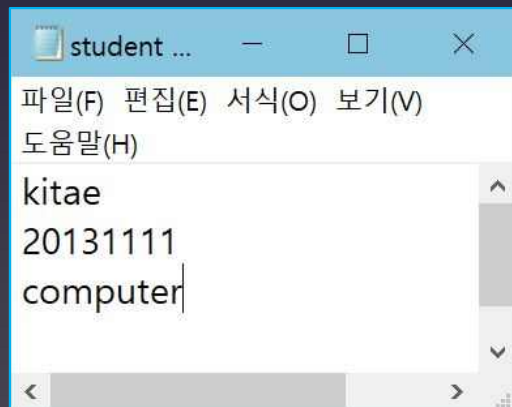
```
// 파일 열기. student.txt 파일을 열고, 출력 스트림 생성
ofstream fout("c:\\temp\\student.txt");
if(!fout) { // 열기 실패 검사
    cout << "c:\\temp\\student.txt 파일을 열 수 없다";
    return 0;
}
```

```
// 파일 쓰기
fout << name << endl; // name 쓰기
fout << sid << endl; // sid 쓰기
fout << dept << endl; // dept 쓰기
```

정수 sid가 문자열로
변환되어 저장됨

```
fout.close(); // 파일 닫기
}
```

이름>>kitae
학번>>20131111
학과>>computer



computer

예제

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    // 스트림 객체 생성 및 파일 열기
    ifstream fin;
    fin.open("c:\\temp\\student.txt");
    if(!fin) { // 파일 열기 실패 확인
        cout << "파일을 열 수 없다";
        return 0;
    }

    char name[10], dept[20];
    int sid;

    // 파일 읽기
    fin >> name; // 파일에 있는 문자열을 읽어서 name 배열에 저장
    fin >> sid; // 정수를 읽어서 sid 정수형 변수에 저장
    fin >> dept; // 문자열을 읽고 dept 배열에 저장

    // 읽은 텍스트를 화면에 출력
    cout << name << endl;
    cout << sid << endl;
    cout << dept << endl;

    fin.close(); // 파일 읽기를 마치고 파일을 닫는다.
}
```

파일의 경로명이 틀리거나 존재하지 않는
파일을 열려고 하면 열기가 실패한다.

kitae
20131111
computer

파일 모드(file mode)

■ 파일 모드란?

- 파일 입출력에 대한 구체적인 작업 행태에 대한 지정
 - 파일에서 읽을 작업을 할 것인지, 쓰기 작업을 할 것인지
 - 기존 파일의 데이터를 모두 지우고 쓸 것인지, 파일의 끝 부분에 쓸 것인지
 - 텍스트 I/O 방식인지 바이너리 I/O 방식 인지

■ 파일 모드 지정 - 파일 열 때

- `open("파일이름", 파일모드)`
- `ifstream("파일이름", 파일모드),`
- `ofstream("파일이름", 파일모드)`

파일 모드	의미
<code>ios::in</code>	읽기 위해 파일을 연다.
<code>ios::out</code>	쓰기 위해 파일을 연다.
<code>ios::ate</code>	(at end) 쓰기 위해 파일을 연다. 열기 후 파일 포인터를 파일 끝에 둔다. 파일 포인터를 옮겨 파일 내의 임의의 위치에 쓸 수 있다.
<code>ios::app</code>	파일 쓰기 시에만 적용된다. 파일 쓰기 시마다, 자동으로 파일 포인터가 파일 끝으로 옮겨져서 항상 파일의 끝에 쓰기가 이루어진다.
<code>ios::trunc</code>	파일을 열 때, 파일이 존재하면 파일의 내용을 모두 지워 파일 크기가 0인 상태로 만든다. <code>ios::out</code> 모드를 지정하면 디폴트로 함께 지정된다.
<code>ios::binary</code>	바이너리 I/O로 파일을 연다. 이 파일 모드가 지정되지 않으면 디폴트가 텍스트 I/O이다.

파일 모드 설정

```
void open(const char * filename, ios::openmode mode)
```

mode로 지정된 파일 모드로 filename의 파일을 연다.

- student.txt 파일에서 처음부터 읽고자 하는 경우

```
ifstream fin;  
fin.open("student.txt");
```

```
ifstream fin;  
fin.open("student.txt", ios::in);
```

- student.txt 파일의 끝에 데이터를 저장하는 경우

```
ofstream fout;  
fout.open("student.txt", ios::out | ios::app);  
fout << "tel:0104447777"; // 기존의 student.txt 끝에 "tel:0104447777"을 추가하여 저장
```

- 바이너리 I/O로 data.bin 파일을 기록하는 경우

```
fstream fbinout;  
fbinout.open("data.bin", ios::out | ios::binary);  
char buf[128];  
....  
fbinout.write(buf, 128); // buf에 있는 128 바이트를 파일에 기록
```

예제 12-3 get()을 이용한 텍스트 파일 읽기

get()을 이용하여 텍스트 파일 c:\windows\system.ini를 읽어 화면에 출력하라.

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    const char* file = "c:\\windows\\system.ini";

    ifstream fin(file);
    if(!fin) {
        cout << file << " 열기 오류" << endl;
        return 0;
    }
    int count = 0;
    int c;
    while((c=fin.get()) != EOF) { // EOF를 만날 때까지 문자 읽기
        cout << (char)c;
        count++;
    }
    cout << "읽은 바이트 수는 " << count << endl;
    fin.close(); // 파일 닫기
}
```

파일에서 문자 읽기

```
; for 16-bit app support
[386Enh]
woafont=dosapp.fon
EGA80WOA.FON=EGA80WOA.FON
EGA40WOA.FON=EGA40WOA.FON
CGA80WOA.FON=CGA80WOA.FON
CGA40WOA.FON=CGA40WOA.FON
```

```
[drivers]
wave=mmdrv.dll
timer=timer.drv
```

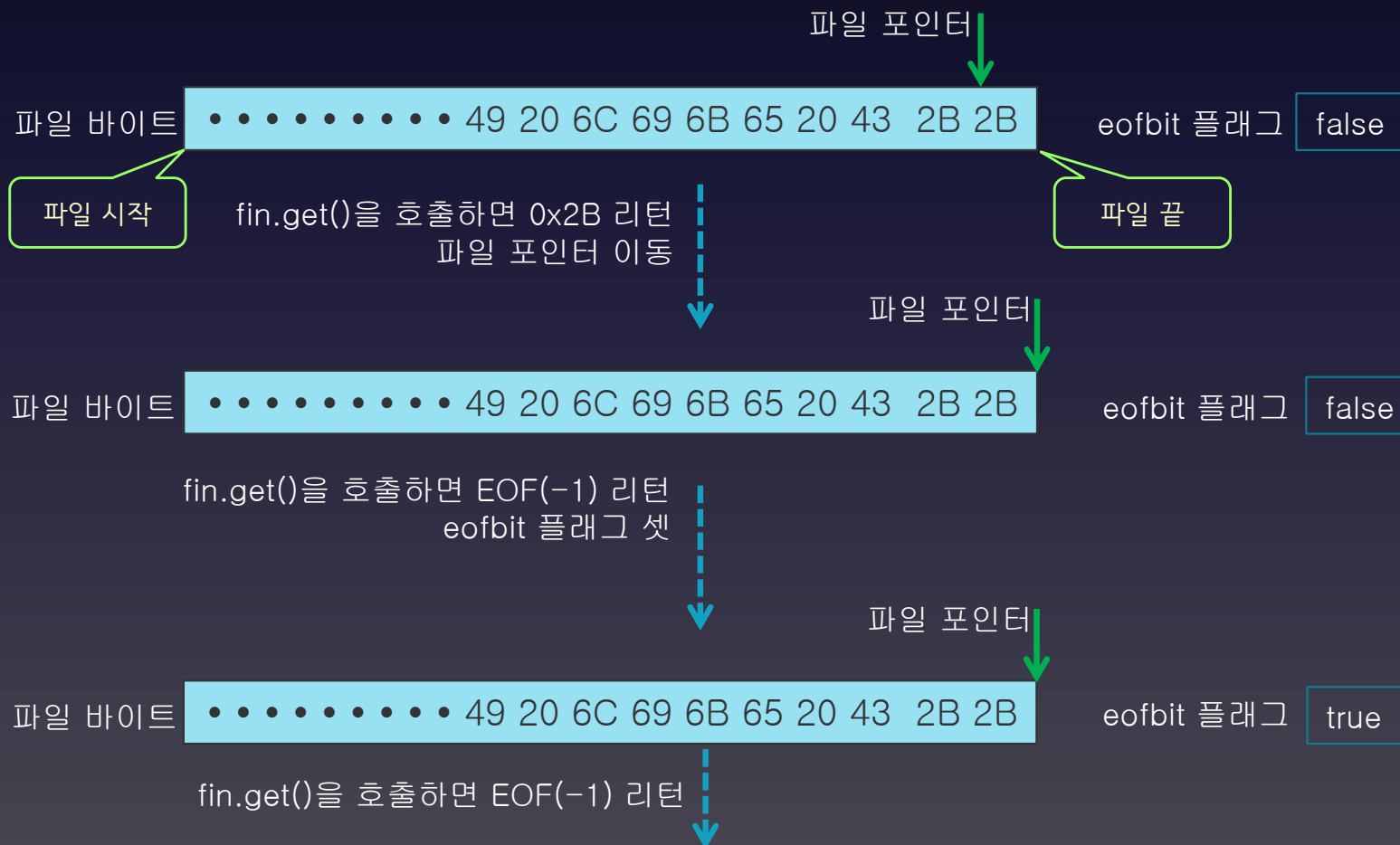
```
[mci]
읽은 바이트 수는 206
```

예제 12-8에서는 219
바이트로 카운트 된다.

텍스트 I/O 모드로 읽을 때, get() 은 라인의 끝에 있는 'WrWn'의
두 바이트를 'Wn'의 한 바이트로 리턴한다.
c:\\windows\\system.ini는 총 13 라인의 219 바이트이지만, 실
제 읽은 바이트 수는 각 라인의 'Wr' 개수 만큼 13개 모자란 206
으로 카운트 된다.

get()과 EOF

- get()은 파일의 끝을 어떻게 인식할까?
 - 파일의 끝에서 읽기를 시도하면 get()은 EOF(-1값)를 리턴한다.



get()으로 파일의 끝을 인지하는 방법

```
while(true) {  
    int c = fin.get(); // 파일에서 문자(바이트)를 읽는다.  
    if(c == EOF) {  
        ..... // 파일의 끝을 만난 경우. 이에 대응하는 코드를 작성  
        break; // while 루프에서 빠져나온다.  
    }  
    else {  
        ..... // 읽은 문자(바이트) c를 처리한다.  
    }  
}
```

|| 동일한 코드

```
while((c = fin.get()) != EOF) { // 파일의 끝을 만나면 루프 종료  
    ..... // 파일에서 읽은 값 c를 처리하는 코드  
}
```


파일의 끝을 잘못 인지하는 코드

```
while(!fin.eof()) {  
    int c = fin.get(); // 마지막 읽은 EOF(-1) 값이 c에 리턴된다.  
    ..... // 읽은 값 c를 처리하는 코드  
}
```

EOF 값을 c에 읽어 사용한 후 다음 루프의 while
조건문에서 EOF에 도달한 사실을 알게 된다

예제 - 텍스트 파일 연결

- `fstream`을 이용하여 `c:\temp\student.txt` 파일에 `c:\windows\system.ini`를 덧붙이는 프로그램을 작성하라.

```
:
#include <fstream>

int main() {
    const char* firstFile = "c:\\temp\\student.txt";
    const char* secondFile = "c:\\windows\\system.ini";

    fstream fout(firstFile, ios::out | ios::app); // 쓰기 모드로 파일 열기
    if(!fout) { // 열기 실패 검사
        cout << firstFile << " 열기 오류";
        return 0;
    }

    fstream fin(secondFile, ios::in); // 읽기 모드로 파일 열기
    if(!fin) { // 열기 실패 검사
        cout << secondFile << " 열기 오류";
        return 0;
    }

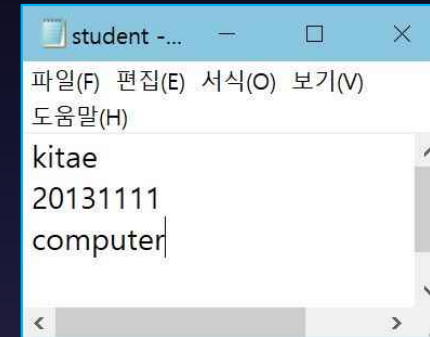
    int c;
    while((c=fin.get()) != EOF) { // 파일 끝까지 문자 읽기
        fout.put(c); // 읽은 문자 기록
    }

    fin.close(); // 입력 파일 닫기
    fout.close(); // 출력 파일 닫기
}
```

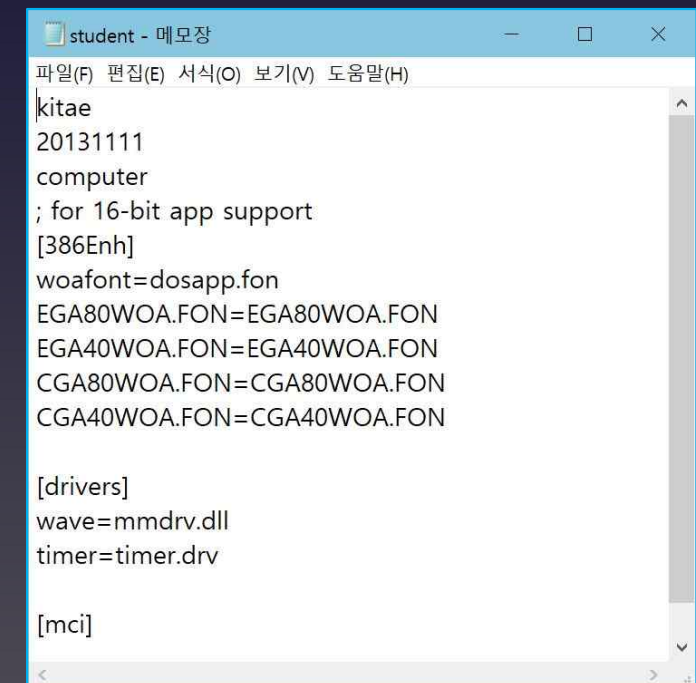
c:\\temp\\student.txt를
덧붙여 쓰기 모드로 열기

c:\\windows\\system.ini를
읽기 모드로 열기

변경된 c:\\temp\\student.txt 파일



원본 c:\\temp\\
student.txt 파일



텍스트 파일의 라인 단위 읽기

■ 두 가지 방법

- istream의 `getline(char* line, int n)` 함수 이용
- `getline(istream& fin, string& line)` 함수 이용

* 라인 단위로 텍스트 파일을 읽는 전형적인 코드

(1) istream의 `getline()` 함수 이용

```
char buf[81];           // 한 라인이 최대 80개의 문자로 구성된다고 가정
ifstream fin("c:\\windows\\system.ini");
while(fin.getline(buf, 81)) { // 한 라인이 최대 80개의 문자로 구성. 끝에 '\0' 문자 추가
    ... // 읽은 라인(buf[])을 활용하는 코드
}
```

(2) 전역 함수 `getline(istream& fin, string& line)` 함수 이용

```
string line;
ifstream fin("c:\\windows\\system.ini");
while(getline(fin, line)) { // 한 라인을 읽어 line에 저장. 파일 끝까지 반복
    ...                     // 읽은 라인(line)을 활용하는 코드 작성
}
```

예제

- c:\Windows\system.ini 파일을 istream의 getline() 함수를 이용하여 한 줄 단위로 읽어 화면에 출력하라

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    ifstream fin("c:\\Windows\\system.ini");
    if(!fin) {
        cout << "c:\\Windows\\system.ini 열기 실패" << endl;
        return 0;
    }

    char buf[81]; // 한 라인이 최대 80개의 문자로 구성된다고 가정
    // 한 라인이 최대 80개의 문자로 구성
    // 라인 출력

    fin.close();
}
```

```
; for 16-bit app support
[386Enh]
woafont=dosapp.fon
EGA80WOA.FON=EGA80WOA.FON
EGA40WOA.FON=EGA40WOA.FON
CGA80WOA.FON=CGA80WOA.FON
CGA40WOA.FON=CGA40WOA.FON

[drivers]
wave=mmdrv.dll
timer=timer.drv

[mci]
```

예제

words.txt 파일을 읽었습니다.
검색할 단어를 입력하세요 >>love
belove
clove
cloven
foxglove
glove
love
lovebird
lovelorn
plover
pullover
sloven
Slovenia
검색할 단어를 입력하세요 >>exit
프로그램을 종료합니다.

love 문자열을 포함하는 단어들

exit을 입력하면
프로그램 종료

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
using namespace std;
```

```
void fileRead(vector<string> &v, ifstream &fin) { // fin으로부터 벡터 v에 읽어 들임
    string line;
    while(getline(fin, line)) { // fin 파일에서 한 라인 읽기
        v.push_back(line); // 읽은 라인을 벡터에 저장
    }
}
```

```
void search(vector<string> &v, string word) { // 벡터 v에서 word를 찾아 출력
    for(int i=0; i<v.size(); i++) {
        int index = v[i].find(word);
        if(index != -1) // found
            cout << v[i] << endl;
    }
}
```

v[i] 단어가 word의 문자열을 포함하는지 검사. -1이 리턴되면 포함하지 않음

```
int main() {
    vector<string> wordVector;
    ifstream fin("words.txt");
    if(!fin) {
        cout << "words.txt 파일을 열 수 없습니다" << endl;
        return 0; // 열기 오류
    }
    fileRead(wordVector, fin); // 파일 전체를 wordVector에 라인 별로 읽기
    fin.close();

    cout << "words.txt 파일을 읽었습니다." << endl;
    while(true) {
        cout << "검색할 단어를 입력하세요 >>";
        string word;
        getline(cin, word); // 키보드로부터 문자열 읽기
        if(word == "exit")
            break; // 프로그램 종료
        search(wordVector, word); // 벡터에서 문자열을 검색하여 출력
    }
    cout << "프로그램을 종료합니다." << endl;
}
```

words.txt 파일이 소스 파일
과 같은 폴더에 있음

바이너리 I/O

■ 바이너리 I/O 방식

- 데이터의 바이너리 값을 그대로 파일에 저장하거나, 파일의 바이너리 값을 그대로 읽어서 변수나 버퍼에 저장하는 방식
- 텍스트 파일이든 바이너리 파일이든 바이너리 I/O로 입출력가능

■ 바이너리 I/O 모드 열기

- `ios::binary` 모드 속성 사용
 - `ios::binary`가 설정되지 않으면 디폴트가 텍스트 I/O

```
ifstream fin;  
fin.open("desert.jpg", ios::in | ios::binary); // 바이너리 I/O로 파일 읽기  
  
ofstream fout("desert.jpg", ios::out | ios::binary); // 바이너리 I/O로 파일 쓰기  
fstream fsin("desert.jpg", ios::in | ios::binary) // 바이너리 I/O로 파일 읽기
```

예제

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    // 소스 파일과 목적 파일의 이름
    const char* srcFile = "c:\\Wtemp\\Wdesert.jpg";
    const char* destFile = "c:\\Wtemp\\Wcopydesert.jpg";

    // 소스 파일 열기
    ifstream fsrc(srcFile, ios::in | ios::binary);
    if(!fsrc) { // 열기 실패 검사
        cout << srcFile << " 열기 오류" << endl;
        return 0;
    }

    // 목적 파일 열기
    ofstream fdest(destFile, ios::out | ios::binary);
    if(!fdest) { // 열기 실패 검사
        cout << destFile << " 열기 오류" << endl;
        return 0;
    }

    // 소스 파일에서 목적 파일로 복사하기
    int c;
    while((c=fsrc.get()) != EOF) { // 소스 파일을 끝까지 한 바이트씩 읽는다.
        fdest.put(c); // 읽은 바이트를 파일에 출력한다.
    }
    cout << srcFile << "을 " << destFile << "로 복사 완료" << endl;
    fsrc.close();
    fdest.close();
}
```

원본 desert.jpg의 경로명

c:\\temp\\copydesert.jpg로
복사



c:\\temp에 있는 desert.jpg



복사된 c:\\temp\\copydesert.jpg

c:\\temp\\desert.jpg을
c:\\temp\\copydesert.jpg로 복사 완
료

read()/write()로 블록 단위 파일 입출력

- `get()/put()`
 - 문자 혹은 바이트 단위로 파일 입출력
- `read()/write()`
 - 블록 단위로 파일 입출력

```
istream& read(char* s, int n)
```

파일에서 최대 *n*개의 바이트를 배열 *s*에 읽어 들임. 파일의 끝을 만나면 읽기 중단

```
ostream& write(char* s, int n)
```

배열 *s*에 있는 처음 *n*개의 바이트를 파일에 저장

```
int gcount()
```

최근에 파일에서 읽은 바이트 수 리턴

예제

- `read()`를 이용하여 한번에 32바이트씩 `c:\windows\system.ini` 파일을 읽어 화면에 출력하는 프로그램을 작성하라.

```
:
#include <fstream>

int main() {
    const char* file = "c:\\windows\\system.ini";

    ifstream fin;
    fin.open(file, ios::in | ios::binary); // 읽기 모드로 파일 열기
    if(!fin) { // 열기 실패 검사
        cout << "파일 열기 오류";
        return 0;
    }

    int count = 0;
    char s[32];
    while(!fin.eof()) { // 파일 끝까지 읽는다.
        fin.read(s, 32); // 최대 32 바이트를 읽어 배열 s에 저장
        int n = fin.gcount(); // 실제 읽은 바이트 수 알아냄
        cout.write(s, n); // 버퍼에 있는 n 개의 바이트를 화면에 출력
        count += n;
    }

    cout << "읽은 바이트 수는 " << count << endl;
    fin.close(); // 입력 파일 닫기
}
```

```
; for 16-bit app support
[386Enh]
woafont=dosapp.fon
EGA80WOA.FON=EGA80WOA.FON
EGA40WOA.FON=EGA40WOA.FON
CGA80WOA.FON=CGA80WOA.FON
CGA40WOA.FON=CGA40WOA.FON
```

```
[drivers]
wave=mmdrv.dll
timer=timer.drv
```

```
[mci]
읽은 바이트 수는 219
```

파일의 크기는 219 바이트임

예제 12-9 read()/write()로 이미지 파일 복사

- read()와 write()를 이용하여 텍스트 파일이든 바이너리 파일이든 복사하는 프로그램을

```
#include <iostream>
#include <fstream>
using namespace std;
```

```
int main() {
    const char* srcFile = "c:\\Wtemp\\tulips.jpg";
    const char* destFile = "c:\\Wtemp\\Wcopytulips.jpg";

    ifstream fsrc(srcFile, ios::in | ios::binary);
    if(!fsrc) { // 열기 실패 검사
        cout << srcFile << " 열기 오류" << endl;
        return 0;
    }
    ofstream fdest(destFile, ios::out | ios::binary);
    if(!fdest) { // 열기 실패 검사
        cout << destFile << " 열기 오류" << endl;
        return 0;
    }
    // 소스 파일에서 목적 파일로 복사하기
    char buf[1024];
    while(!fsrc.eof()) { // 파일 끝까지 읽는다.
        fsrc.read(buf, 1024); // 최대 1024 바이트를 읽어 배열 s에 저장
        int n = fsrc.gcount(); // 실제 읽은 바이트 수 알아냄
        fdest.write(buf, n); // 읽은 바이트 수 만큼 버퍼에서 목적 파일에 기록
    }
    cout << srcFile << "을 " << destFile << "로 복사 완료" << endl;
    fsrc.close();
    fdest.close();
}
```

c:\Wtemp 폴더의 tulips.jpg의 경로명

c:\Wtemp\Wcopytulips.jpg로
복사



c:\Wtemp 폴더에 있는 tulips.jpg



복사된 c:\Wtemp\Wcopytulips.jpg

c:\Wtemp\tulips.jpg를
c:\Wtemp\Wcopytulips.jpg로 복사 완
료

예제

```
#include <iostream>
#include <fstream>
using namespace std;
```

```
int main() {
    char* file = "c:\\temp\\data.dat";
```

바이너리 I/O 모드 설정

```
    ofstream fout;
    fout.open(file, ios::out | ios::binary); // 읽기 모드로 파일 열기
    if(!fout) { // 열기 실패 검사
        cout << "파일 열기 오류";
        return 0;
    }
```

```
    int n[] = {0,1,2,3,4,5,6,7,8,9};
    double d = 3.15;
    fout.write((char*)n, sizeof(n)); // int 배열 n을 한번에 파일에 쓴다.
    fout.write((char*)&d, sizeof(d)); // double 값 하나를 파일에 쓴다.
    fout.close();
```

write()로 한번에 배열을 쓴다.

```
    // 배열 n과 d 값을 임의의 값으로 변경시킨다.
    for(int i=0; i<10; i++) n[i]=99;
    d = 8.15;
```

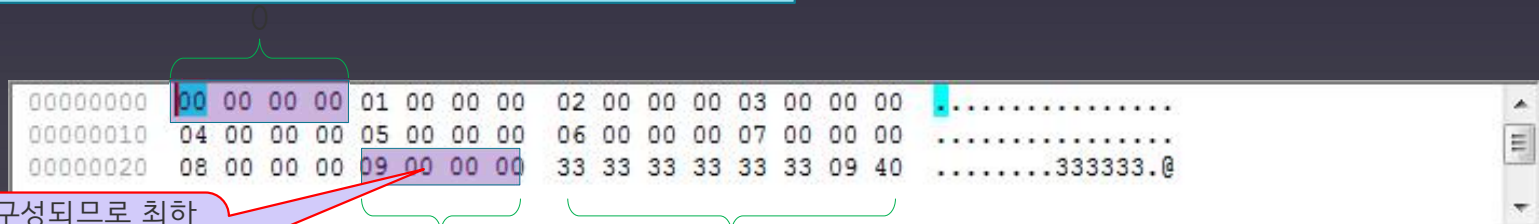
```
    // 배열 n과 d 값을 파일에서 읽어 온다.
    ifstream fin(file, ios::in | ios::binary);
    if(!fin) { // 열기 실패 검사
        cout << "파일 열기 오류";
        return 0;
    }
```

read()로 한번에 배열을 읽는다.

```
    fin.read((char*)n, sizeof(n));
    fin.read((char*)&d, sizeof(d));

    for(int i=0; i<10; i++)
        cout << n[i] << ' ';
    cout << endl << d << endl;
    fin.close();
}
```

```
0 1 2 3 4 5 6 7 8 9
3.15
```

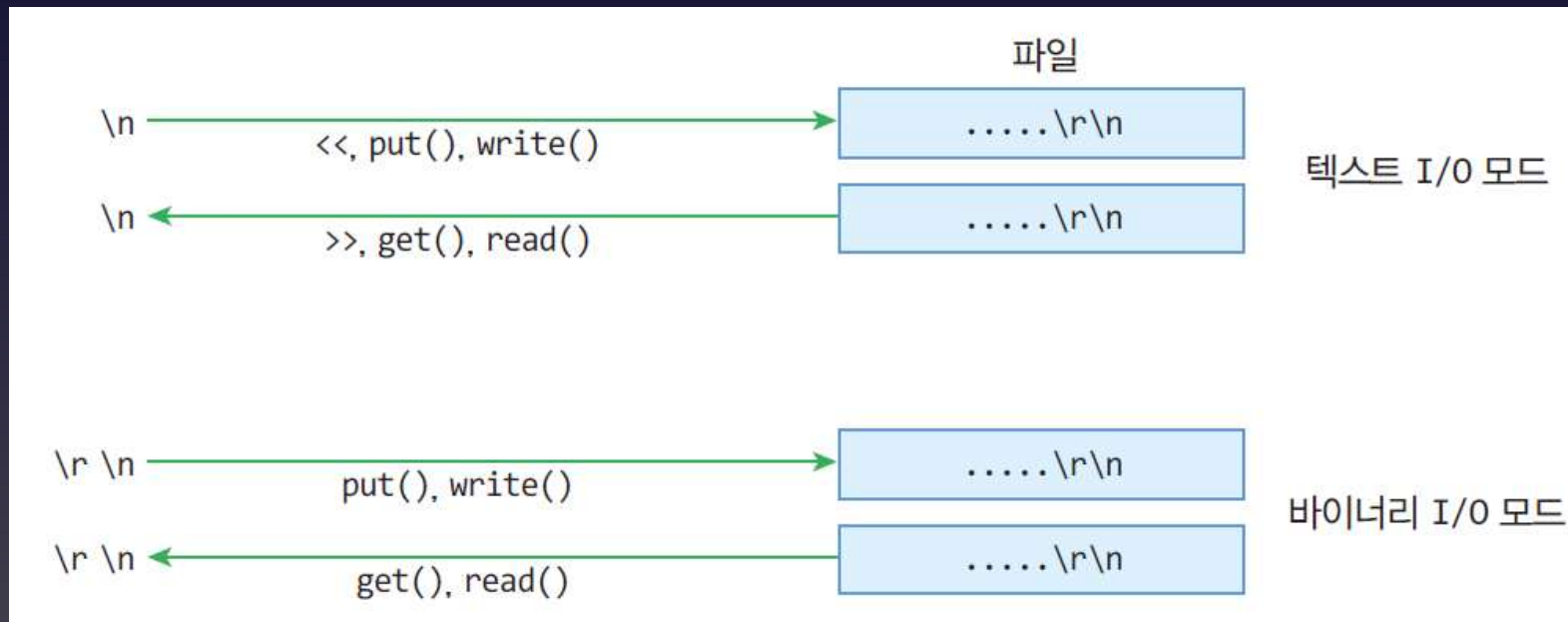


int 타입의 정수는 4 바이트로 구성되므로 최하위 바이트부터 4 바이트를 기록한다. 그러므로 09 00 00 00의 4 바이트는 거꾸로 조합하면 00000009의 값이다.

c:\\temp\\data.dat 파일 내부(바이너리 파일)

텍스트 I/O와 바이너리 I/O의 확실한 차이점

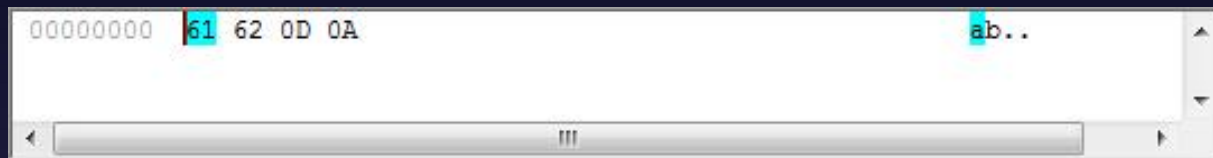
- 파일의 끝을 처리하는 방법에는 차이가 없다.
 - 텍스트 I/O 든 바이너리 I/O 든 파일의 끝을 만나면 EOF 리턴
- 개행 문자 '\n'를 읽고 쓸 때 서로 다르게 작동한다.



텍스트 I/O와 바이너리 I/O의 실행 결과 비교

텍스트 I/O 모드

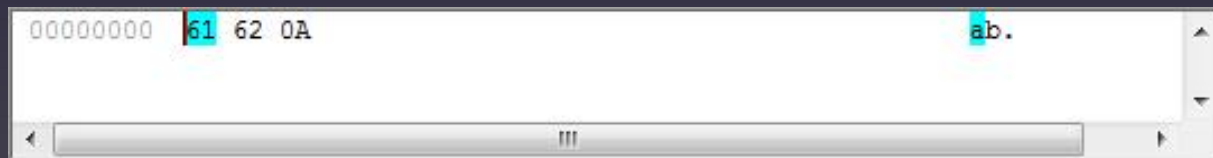
```
char buf[] = {'a', 'b', '\n'};  
fout.write(buf, 3);  
// 파일에 'a', 'b', '\r', '\n'의 4 개의 바이트 저장
```



A hex editor window showing the memory contents after a text I/O operation. The address 00000000 is displayed on the left. The hex value 61 62 0D 0A is shown, with '61' highlighted in blue. To the right, the corresponding ASCII characters 'ab..' are displayed, with 'ab' highlighted in blue. A scrollbar is visible at the bottom.

바이너리 I/O 모드

```
ofstream fout("c:\\student3.txt", ios::out | ios::binary);  
char buf[] = {'a', 'b', '\n'};  
fout.write(buf, 3);  
// 파일에 'a', 'b', '\n'의 3 개의 바이트 저장
```

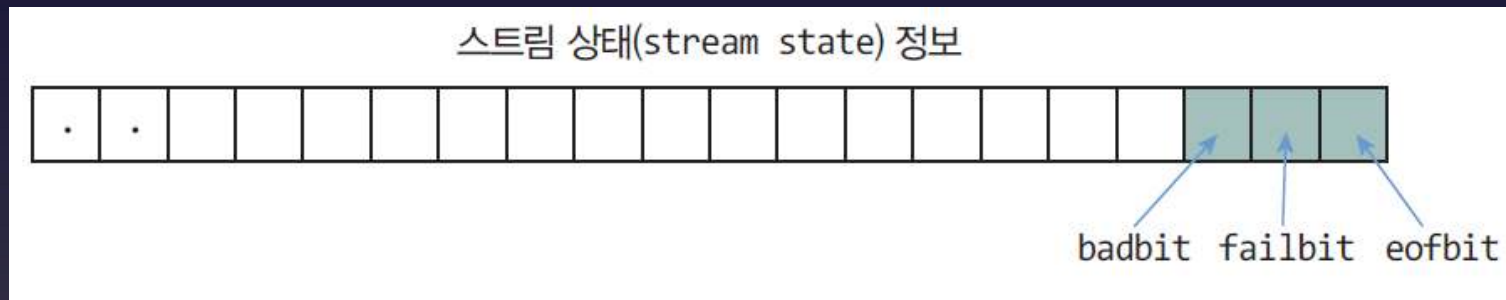


A hex editor window showing the memory contents after a binary I/O operation. The address 00000000 is displayed on the left. The hex value 61 62 0A is shown, with '61' highlighted in blue. To the right, the corresponding ASCII characters 'ab.' are displayed, with 'ab.' highlighted in blue. A scrollbar is visible at the bottom.

스트림 상태 검사

□ 스트림 상태

- ▣ 파일 입출력이 진행되는 동안 스트림(열어 놓은 파일)에 관한 입출력 오류 저장
 - 스트림 상태를 저장하는 멤버 변수 이용



스트림 상태

비트	설명
eofbit	파일의 끝을 만났을 때 1로 세팅
failbit	정수를 입력받고자 하였으나 문자열이 입력되는 등 포맷 오류나, 쓰기 금지된 곳에 쓰기를 시행하는 등 전반적인 I/O 실패 시에 1로 세팅
badbit	스트림이나 데이터가 손상되는 수준의 진단되지 않는 문제가 발생한 경우나 유효하지 않는 입출력 명령이 주어졌을 때 1로 세팅

멤버 함수	설명
eof()	파일의 끝을 만났을 때(eofbit=1) true 리턴
fail()	failbit나 badbit가 1로 세팅되었을 때 true 리턴
bad()	badbit이 1로 세팅되었을 때 true 리턴
good()	스트림이 정상적(모든 비트가 0)일 때 true 리턴
clear()	스트림 상태 변수를 0으로 지움

예제 - 스트림 상태 검사

```
#include <iostream>
#include <fstream>
using namespace std;
```

```
void showStreamState(ios& stream) {
    cout << "eof() " << stream.eof() << endl;
    cout << "fail() " << stream.fail() << endl;
    cout << "bad() " << stream.bad() << endl;
    cout << "good() " << stream.good() << endl;
}
```

```
int main() {
    const char* noExistFile = "c:\\Wtemp\\Wnoexist.txt"; // 존재하지 않는 파일명
    const char* existFile = "c:\\Wtemp\\Wstudent.txt"; // 존재하는 파일명
```

존재하지 않는 파일을 열 때, 스트림의 상태가 어떻게 변하는지 알기 위한 시도

```
ifstream fin(noExistFile); // 존재하지 않는 파일 열기
if(!fin) { // 열기 실패 검사
    cout << noExistFile << " 열기 오류" << endl;
    showStreamState(fin); // 스트림 상태 출력
}
```

정상적인 파일을 열 때, 스트림의 상태가 어떠한지 보기 위한 시도

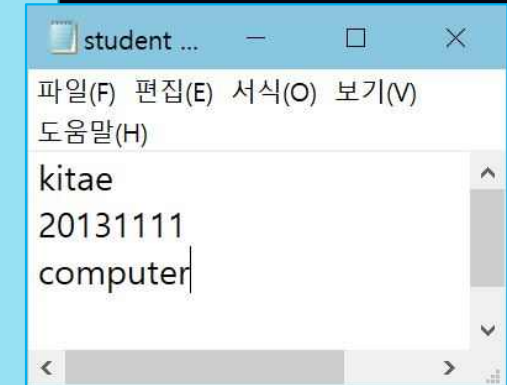
```
cout << existFile << " 파일 열기" << endl;
fin.open(existFile);
showStreamState(fin); // 스트림 상태 출력
}
```

```
// 스트림을 끝까지 읽고 화면에 출력
int c;
while((c=fin.get()) != EOF)
    cout.put((char)c);
```

EOF를 만났을 때, 스트림 상태 출력

```
cout << endl;
showStreamState(fin); // 스트림 출력

fin.close();
}
```



```
c:\\Wtemp\\Wnoexist.txt 열기 오류
eof() 0
fail() 1
bad() 0
good() 0
c:\\Wtemp\\Wstudent.txt 파일 열기
eof() 0
fail() 0
bad() 0
good() 1
kitae
20131111
computer
eof() 1
fail() 1
bad() 0
good() 0
```


임의 접근과 파일 포인터

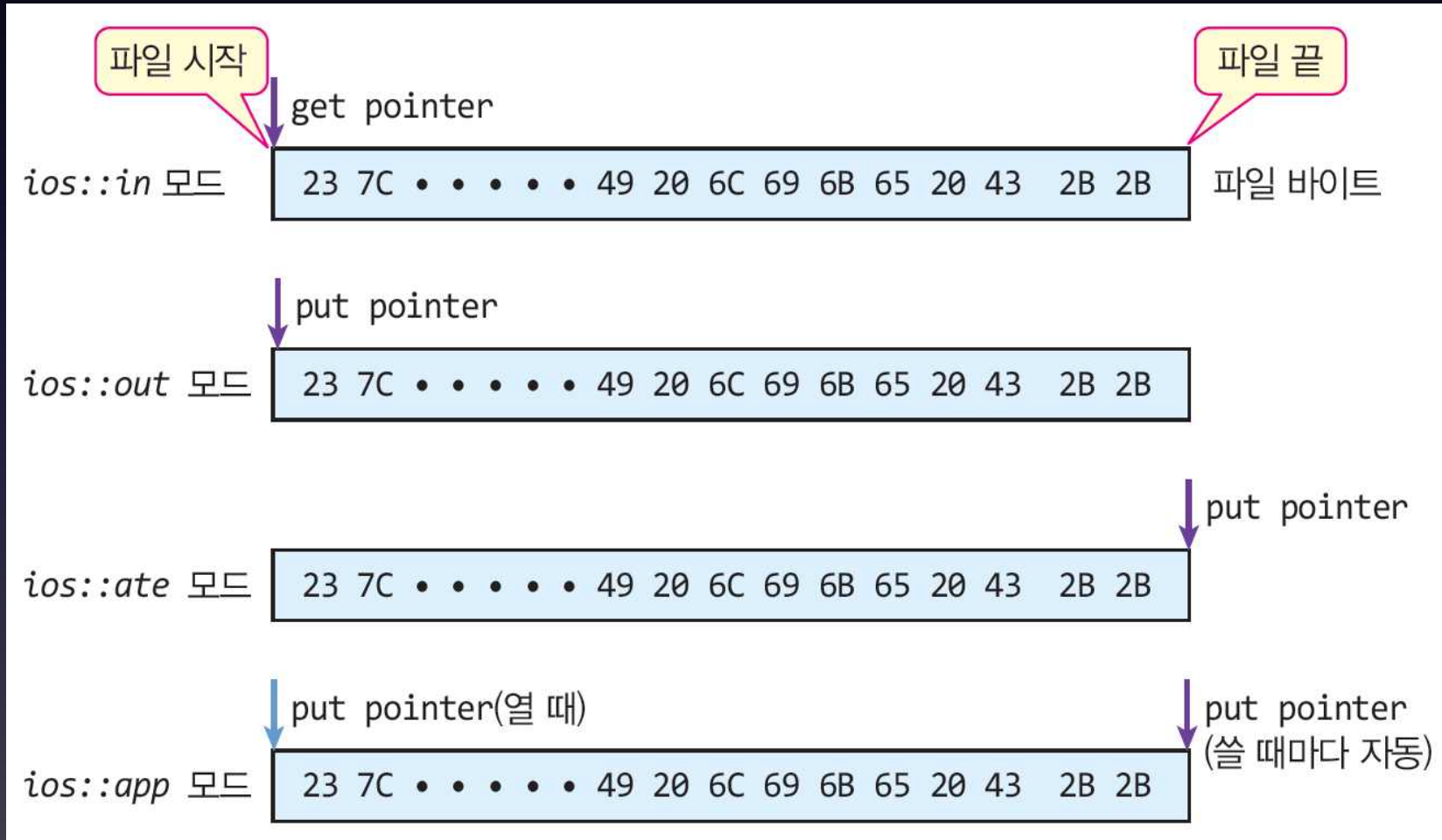
■ C++ 파일 입출력 방식

- 순차 접근
 - 읽은 다음 위치에서 읽고, 쓴 다음 위치에 쓰는 방식
 - 디폴트 파일 입출력 방식
- 임의 접근
 - 파일 내의 임의의 위치로 옮겨 다니면서 읽고 쓸 수 있는 방식
 - 파일 포인터를 옮겨 파일 입출력

■ 파일 포인터(file pointer)

- 파일은 연속된 바이트의 집합
- 파일 포인터
 - 파일에서 다음에 읽거나 쓸 위치를 표시하는 특별한 마크
- C++는 열려진 파일마다 두 개의 파일 포인터 유지
 - get pointer : 파일 내에 다음에 읽을 위치
 - put pointer : 파일 내에 다음에 쓸 위치

파일 모드와 파일 포인터



입력의 접근 방법

■ 파일 포인터 제어

- 절대 위치로 이동시키는 방법과 상대 위치로 이동시키는 두 방법

```
istream& seekg(streampos pos)
```

정수 값으로 주어진 절대 위치 pos로 get pointer를 옮김

```
istream& seekg(streamoff offset, ios::seekdir seekbase)
```

seekbase를 기준으로 offset 만큼 떨어진 위치로 get pointer를 옮김

```
ostream& seekp(streampos pos)
```

정수 값으로 주어진 절대 위치 pos로 put pointer를 옮김

```
ostream& seekp(streamoff offset, ios::seekdir seekbase)
```

seekbase를 기준으로 offset 만큼 떨어진 위치로 put pointer를 옮김

```
streampos tellg()
```

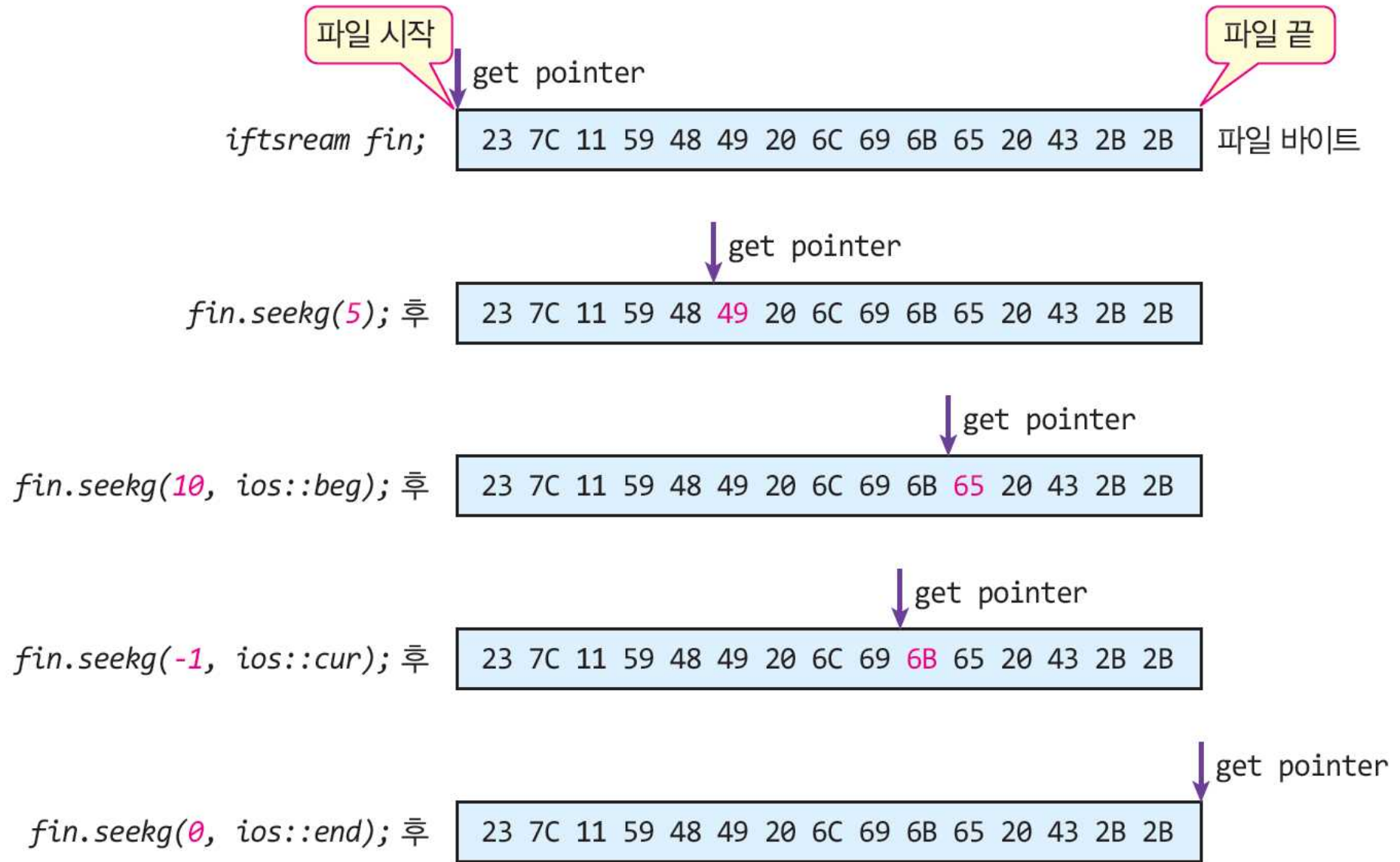
입력 스트림의 현재 get pointer의 값 리턴

```
streampos tellp()
```

출력 스트림의 현재 put pointer의 값 리턴

seekbase	설명
ios::beg	파일의 처음 위치를 기준으로 파일 포인터를 움직인다.
ios::cur	현재 파일 포인터의 위치를 기준으로 파일 포인터를 움직인다.
ios::end	파일의 끝(EOF) 위치를 기준으로 파일 포인터를 움직인다.

seekg()



예제 : 파일 크기 알아내기

- c:\windows\system.ini 파일의 크기가 몇 바이트인지 알아내어 출력하라.

c:\windows\system.ini
파일의 속성 보기 창.
파일 크기가 219바이트
임을 확인

```
#include <iostream>
#include <fstream>
using namespace std;

long getFileSize(ifstream& fin) {
    fin.seekg(0, ios::end); // get pointer를 파일의 맨 끝으로 옮김
    long length = fin.tellg(); // get pointer의 위치를 알아냄
    return length; // length는 파일의 크기와 동일
}

int main() {
    const char* file = "c:\\windows\\system.ini";

    ifstream fin(file);
    if(!fin) { // 열기 실패 검사
        cout << file << " 열기 오류" << endl;
        return 0;
    }
    cout << file << "의 크기는 " << getFileSize(fin);
    fin.close();
}
```

c:\windows\system.ini의 크기는 219



