13장 파일 처리

2020. 12. 7

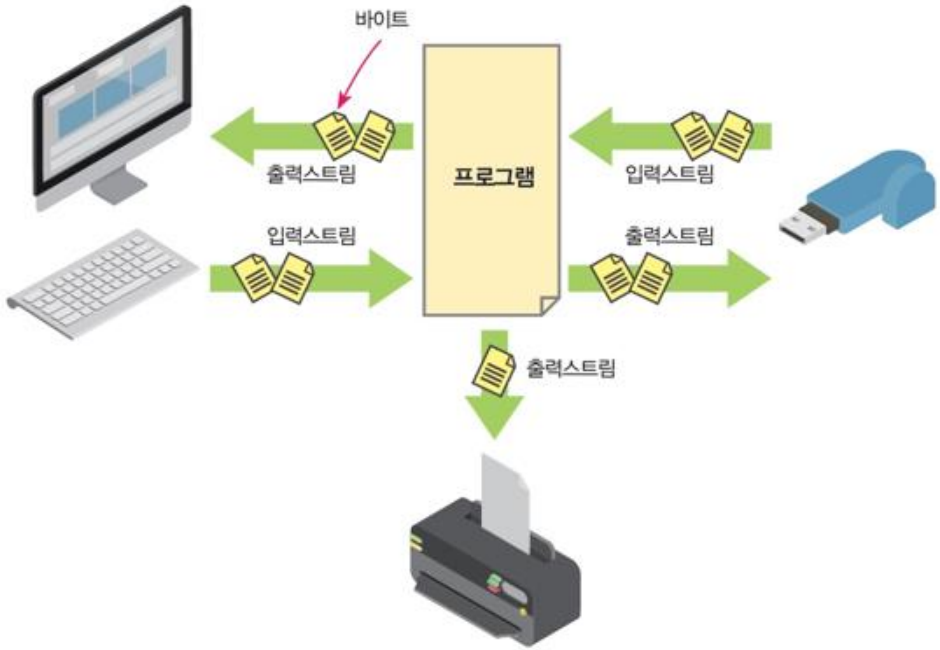
순천향대학교 컴퓨터 공학과

내용

- 스트림
- 파일 출력
- 파일 입력
- 파일 모드
- 출력 형식 지정

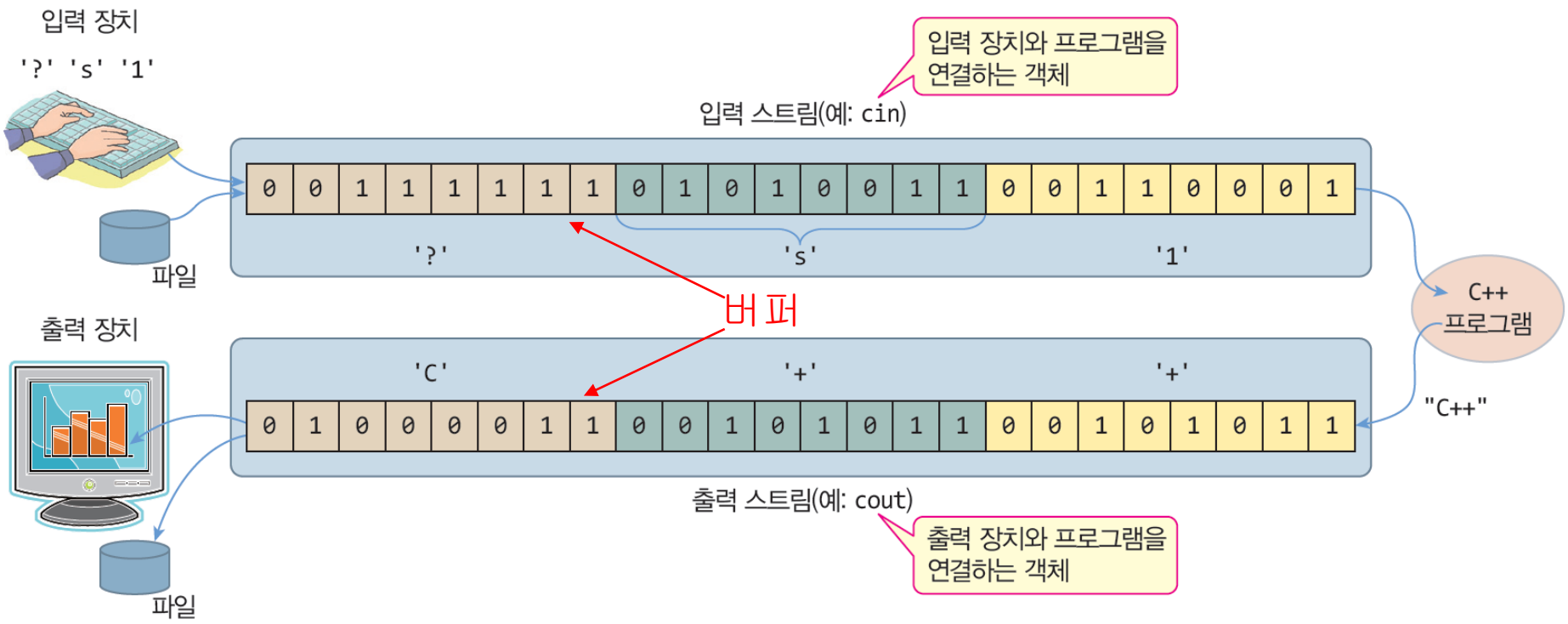
스트림(stream)

- 스트림(stream)
  - "순서가 있는 데이터의 연속적인 흐름"이다.
  - 스트림 양 끝에 프로그램과 장치 연결
  - C++ 스트림 종류: 입력 스트림, 출력 스트림



C++ 입출력 스트림

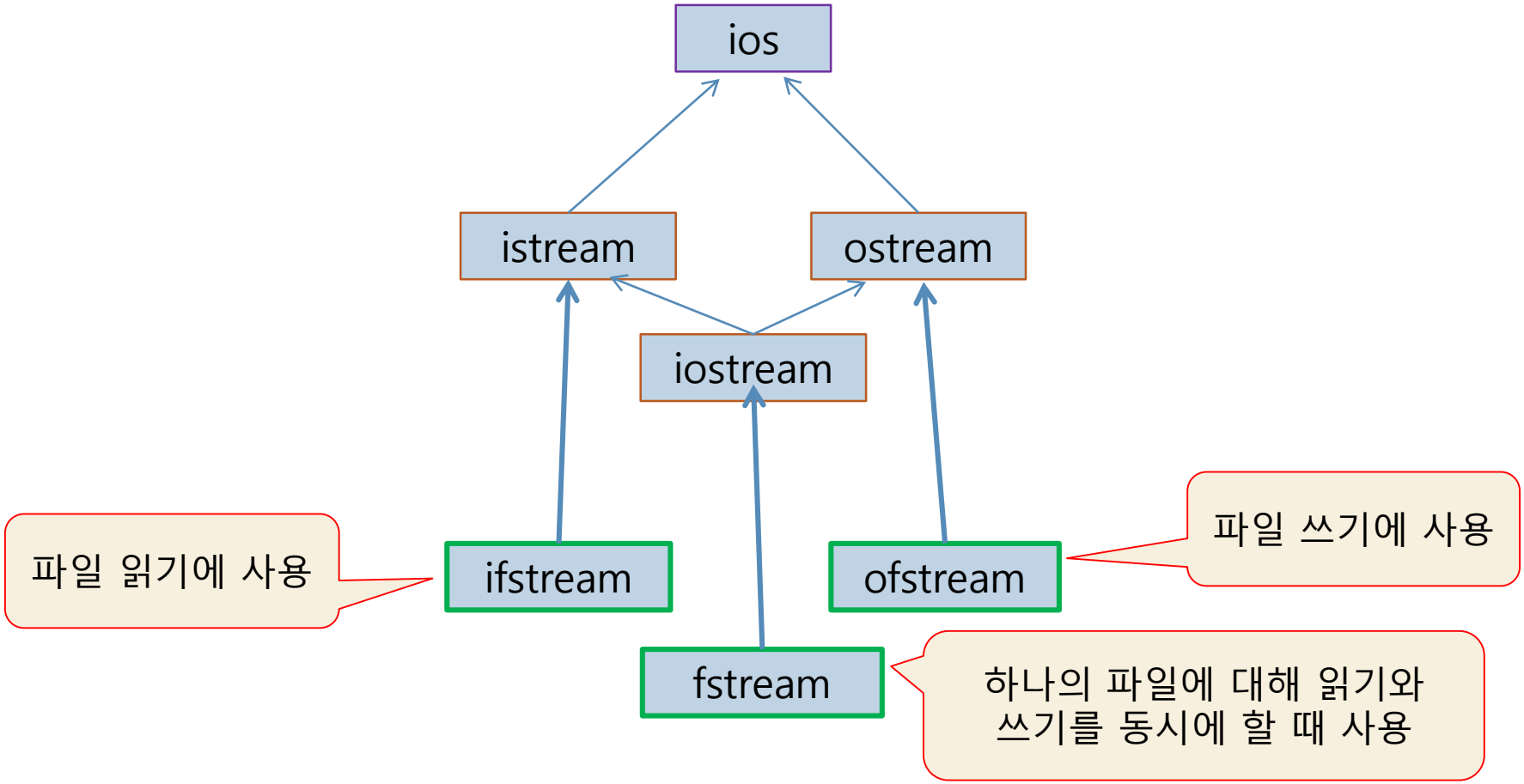
- cin, cout을 이용한 입출력



C++ 입출력 클래스 소개

클래스	설명
ios	모든 입출력 스트림 클래스들의 기본(Base) 클래스. 스트림 입출력에 필요한 공통 함수와 상수, 멤버 변수 선언
istream, ostream, iostream	istream은 문자 단위 입력 스트림. ostream은 문자 단위 출력 스트림. iostream은 문자 단위로 입출력을 동시에 할 수 있는 스트림 클래스
ifstream, ofstream, fstream	파일에서 읽고 쓰는 기능을 가진 파일 입출력 스트림 클래스. 파일에서 읽을 때는 ifstream 클래스를, 파일에 쓸 때는 ofstream 클래스를, 읽고 쓰기를 동시에 할 때 fstream 클래스 이용

C++ 표준 파일 입출력 클래스 관계



표준 입출력 스트림 객체

- C++ 프로그램이 실행될 때 자동 생성되는 스트림 객체
  - cin
    - istream타입의 스트림 객체로서 키보드 장치와 연결
  - cout
    - ostream타입의 스트림 객체로서 스크린 장치와 연결
  - cerr
    - ostream타입의 스트림 객체로서 스크린 장치와 연결
    - 오류 메시지를 출력할 목적

파일 입출력 관련 클래스들

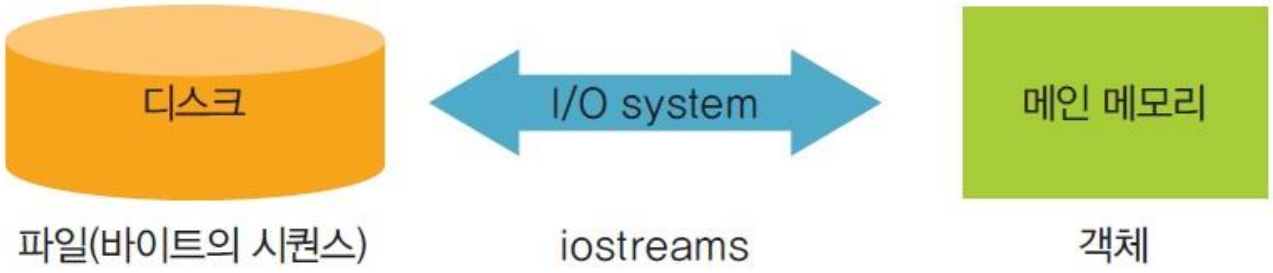


그림 13.2 스트림의 개념

클래스	설명
ofstream	출력 파일 스트림 클래스이다. 출력 파일을 생성하고 파일에 데이터를 쓸때 사용한다.
ifstream	입력 파일 스트림 클래스이다. 파일에서 데이터를 읽을 때 사용한다.
fstream	일반적인 파일 스트림을 나타낸다.

헤더 파일과 namespace

- C++ 파일 입출력(ifstream, ofstream, fstream 사용)을 위한 필요 사항
  - <fstream> 헤더 파일과 std 이름 공간의 선언

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

파일 쓰기

- 형식 1:

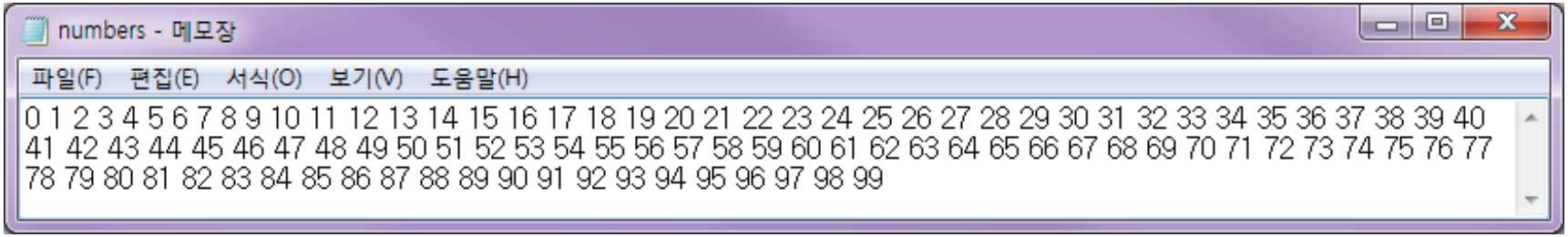
```
ofstream os;                    // 파일출력스트림 객체 생성
os.open("result.txt");          // 파일 열기
...
os.close();                     // 파일 닫기
```

- 형식 2:

```
ofstream os("result.txt"); // 파일출력스트림 객체 생성과
                           // 동시에 파일 열기
...

// 객체가 소멸될 때 자동으로 파일을 닫는다.
```

예제: 파일 쓰기

```
int main()
{
  ofstream os{ "numbers.txt" }; // 파일출력 스트림 생성과 동시에 파일 열기

  if (!os) { // os 객체의 operator!() 연산자 함수가 실행되고, 이 함수는
             // 파일 열기가 성공이면 true, 그렇지 않으면 false 반환
             cerr << "파일 오픈에 실패하였습니다" << endl;
             exit(1);
  }
  for(int i=0;i<100; i++)
             os << i <<" ";
  return 0;

 // os 객체가 소멸되면서 파일을 자동으로 닫는다.
}
```



예제: 파일 읽기

```
int main()
{
        ifstream is{ "numbers.txt" };
        if (!is) {
                cerr << "파일 오픈에 실패하였습니다" << endl;
                exit(1);
        }
        int number;
        while (is) {
                is >> number;
                cout << number << " ";
        }
        cout << endl;
        return 0;

        // 객체 is가 소멸될때  ifstream 소멸자가 파일을 닫는다.
}
```

파일 모드(file mode)

- 파일 모드는 파일을 열 때 입출력 작업 유형 지정
- 형식:
  - open("파일이름", 파일모드) // 객체 생성후 파일 열기
  - ifstream("파일이름", 파일모드), 디폴트는 ios::in
  - ofstream("파일이름", 파일모드), 디폴트는 ios::out

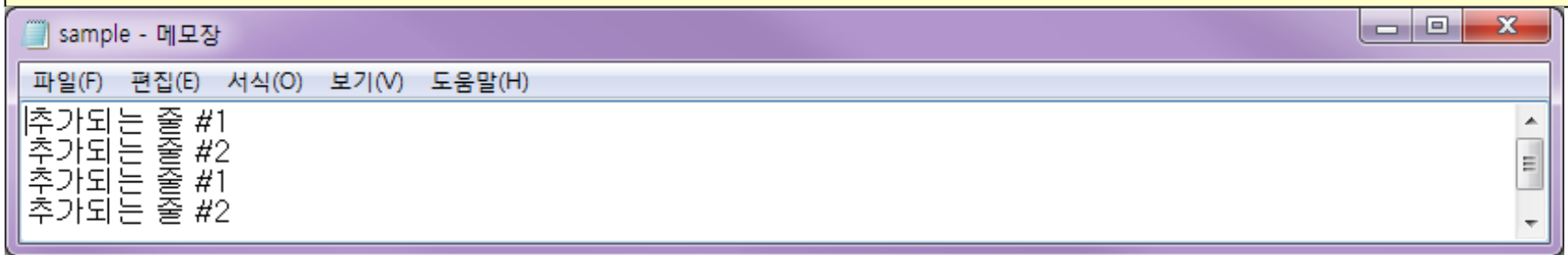
파일 모드

- 파일 모드는 ios 클래스에 선언된 상수

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

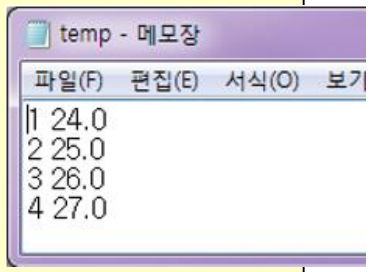
예제

```
int main()
{
        using namespace std;

        ofstream os("sample.txt", ios::app);
        if (!os)
        {
                cerr << "파일 오픈에 실패하였습니다" << endl;
                exit(1);
        }

        os << "추가되는 줄 #1" << endl;
        os << "추가되는 줄 #2" << endl;

        return 0;
}
```



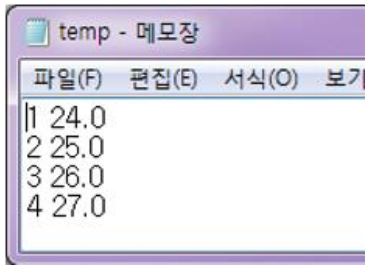
Lab#1: 온도 데이터 처리

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

int main()
{
        ifstream is{ "temp.txt" };
        if (!is) {
                cerr << "파일 오픈에 실패하였습니다" << endl;
                exit(1);
        }

        int hour;
        double temperature;

        while (is >> hour >> temperature) {
                cout << hour << "시: 온도 "<< temperature << endl;
        }
        return 0;
}
```



Lab#2: 온도 데이터 처리

- 파일에서 읽은 데이터를 객체로 벡터에 저장했다가 다시 꺼내서 화면에 출력해보자.



```
class TempData {
public:
        int hour;
        double temperature;
};
int main() {
        ifstream is{ "temp.txt" };
        if (!is) {
                cerr << "파일 오픈에 실패하였습니다" << endl;
                exit(1);
        }
        vector<TempData> temps;
        int hour;
        double temperature;

        while (is >> hour >> temperature) // 벡터에 저장
                temps.push_back(TempData{ hour, temperature });

        for ( TempData t : temps) { // 벡터 요소 출력
                cout << t.hour << "시: 온도 " << t.temperature << endl;
        }
        return 0;
}
```

멤버 함수를 이용한 파일 입출력

- fstream의 멤버 함수
  - get() : 하나의 문자를 입력
  - put(): 한 개의 문자 출력
  - getline(): 한 줄 입력
  - eof(): 파일 끝 탐지

멤버함수를 이용한 파일 입출력

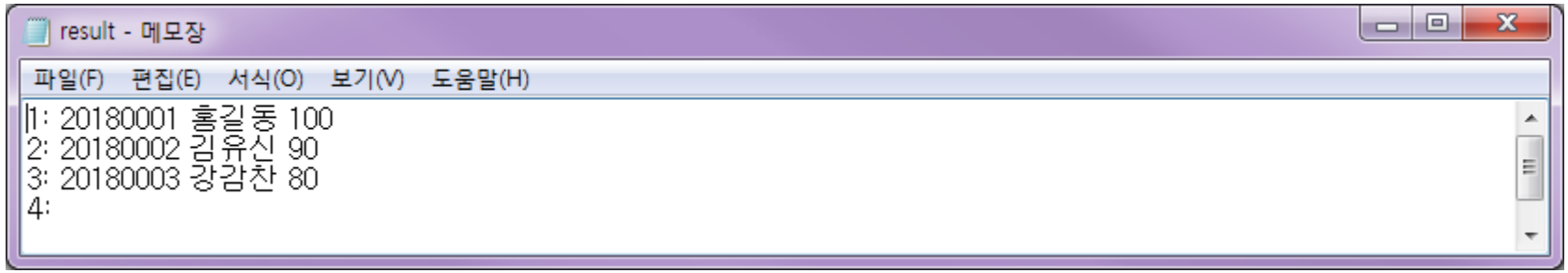
```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
	ifstream is{ "scores.txt" };
	if (!is) {
		cerr << "파일 오픈에 실패하였습니다" << endl;
		exit(1);
	}
	char c;
	is.get(c);		// 하나의 문자를 읽는다.
	while (!is.eof()) {	// 파일의 끝이 아니면
		cout << c;
		is.get(c);
	}
	cout << endl;
	return 0;
}
```

예제

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

int main() {
        string address;
        cout << "주소를 입력하시오: ";
        getline(cin, address);
        cout << "안녕! " << address << "에 사시는 분" << endl;

        return 0;
}
```

Lab: 줄번호

- 소스가 저장된 텍스트 파일을 읽어서 각 줄의 앞에 숫자를 붙인 후에 출력 파일에 기록하여 보자.



```
int main() {
        ifstream is("scores.txt");;
        ofstream os("result.txt"); ;
        if (is.fail()) {
                cerr << "파일 오픈 실패" << endl;
                exit(1);
        }
        if (os.fail()) {
                cerr << "파일 오픈 실패" << endl;
                exit(1);
        }
                char c;

                int line_number = 1;
                is.get(c);
                os << line_number << ": ";
                while (!is.eof()) {
                        os << c;
                        if (c == '\n') {
                                line_number++;
                                os << line_number << ": ";
                        }
                        is.get(c);
                }
                return 0;
}
```

출력 형식 지정 플래그(ios에 정의)

- 표준이나 파일 출력에 출력 형식 지정 가능

플래그	설명
ios::fixed	고정 소수점 표기법 사용
ios::scientific	과학적 표기법 사용(지수를 이용하여 표기)
ios::showpoint	소수점을 반드시 표시한다.
ios::showpos	양수 부호를 반드시 출력한다.
ios::right	값을 출력할 때 오른쪽 정렬을 사용한다.
ios::left	값을 출력할 때 왼쪽 정렬을 사용한다.
ios::dec	값을 출력할 때 10진법을 사용한다.
ios::oct	값을 출력할 때 8진법을 사용한다.
ios::hex	값을 출력할 때 16진법을 사용한다.
ios::uppercase	지수나 16진법으로 표시할 때 대문자를 사용한다.
ios::show	8진수이면 앞에 0을 붙이고 16진수이면 앞에 0x를 붙인다.

포맷 플래그를 세팅하는 멤버 함수

long setf(long flags)

flags를 스트림의 포맷 플래그로 설정하고 이전 플래그를 리턴한다.

long unsetf(long flags)

flags에 설정된 비트 값에 따라 스트림의 포맷 플래그를 해제하고 이전 플래그를 리턴한다.

```
cout.unsetf(ios::dec);      // 10진수 해제
cout.setf(ios::hex);        // 16진수로 설정
cout << 30 << endl;         // 1e가 출력됨
```

```
cout.setf(ios::dec | ios::showpoint); // 10진수 표현과 동시에 실수에
                                      // 소숫점이하 나머지는 0으로 출력
cout << 23.5 << endl; // 23.5000으로 출력
```

```
int main() {
  cout << 30 << endl; // 10진수로 출력

  cout.unsetf(ios::dec); // 10진수 해제
  cout.setf(ios::hex); // 16진수로 설정
  cout << 30 << endl;

  cout.setf(ios::showbase); // 16진수로 설정
  cout << 30 << endl;

  cout.setf(ios::uppercase); // 16진수의 A~F는 대문자로 출력
  cout << 30 << endl;

  cout.setf(ios::dec | ios::showpoint); // 10진수 표현과 동시에
                                        // 소숫점 이하 나머지는 0으로 출력
  cout << 23.5 << endl;

  cout.setf(ios::scientific); // 실수를 과학산술용 표현으로 출력
  cout << 23.5 << endl;

  cout.setf(ios::showpos); // 양수인 경우 + 부호도 함께 출력
  cout << 23.5;
}
```

출력

```
30
1e
0x1e
0X1E
23.5000
2.350000E+001
+2.350000E+001
```