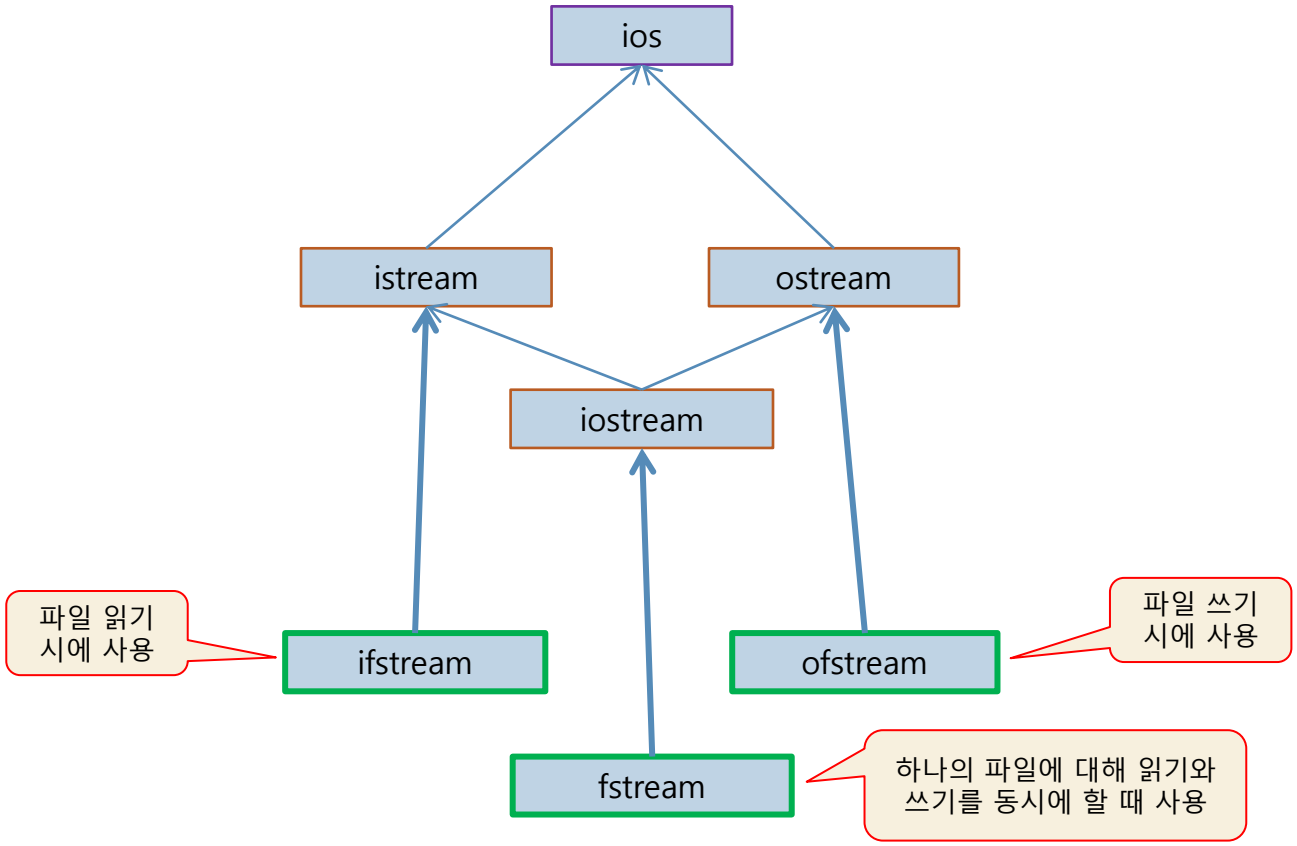
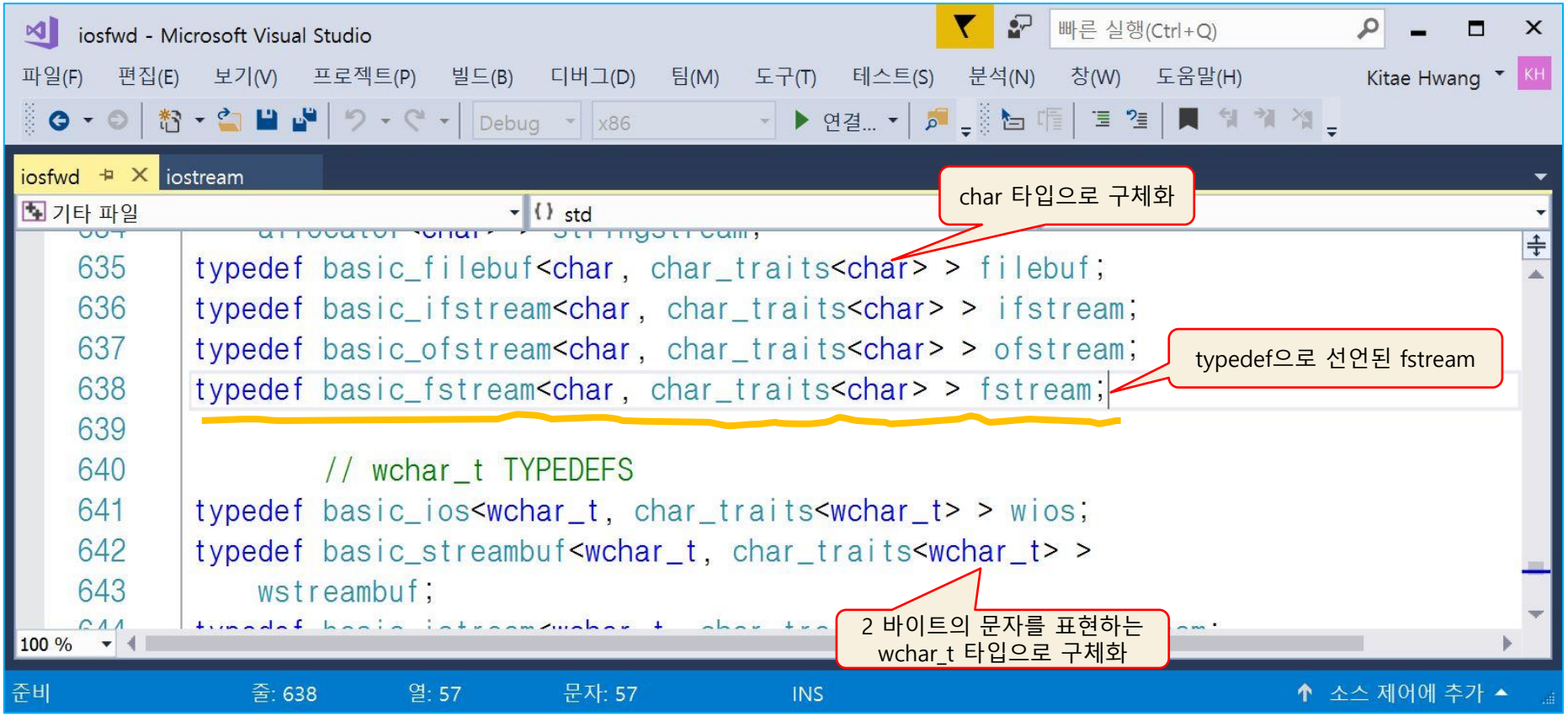
C++ 표준 파일 입출력 라이브러리

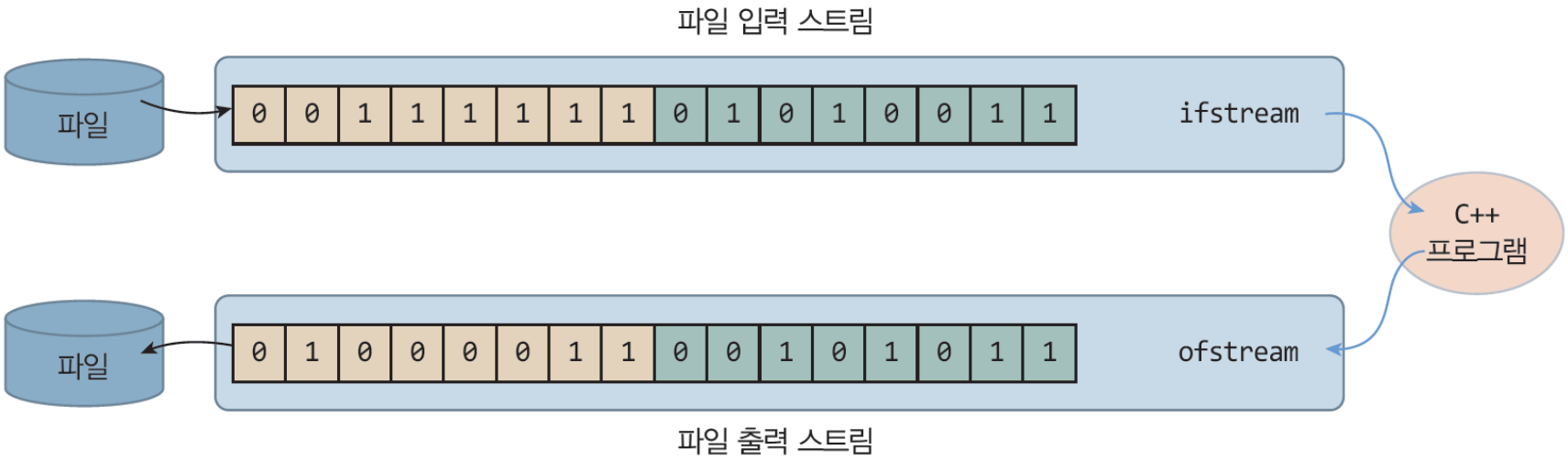
- 스트림 입출력 방식 지원



템플릿에 char 타입으로 구체화한 클래스들



파일 입출력 스트림은 파일을 프로그램과 연결한다.



- >> 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

파일 입출력 모드 : 텍스트 I/O와 바이너리 I/O

- 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

헤더 파일과 namespace

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

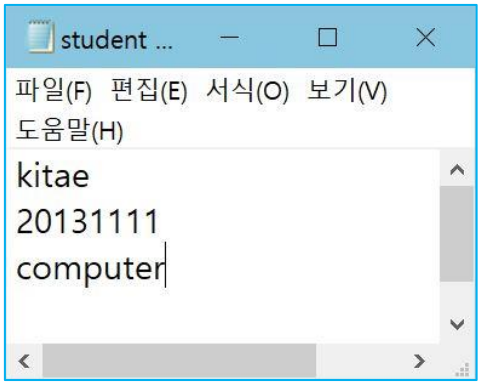
```
#include <fstream>
using namespace std;

스트림 객체 생성
ofstream fout;   출력 스트림 객체 생성
ifstream fin;    입력 스트림 객체 생성

파일 열기
fout.open("a.txt");   출력할 파일 열기
fin.open("a.txt");    입력할 파일 열기

파일 닫기
fout.close();
fin.close();
```

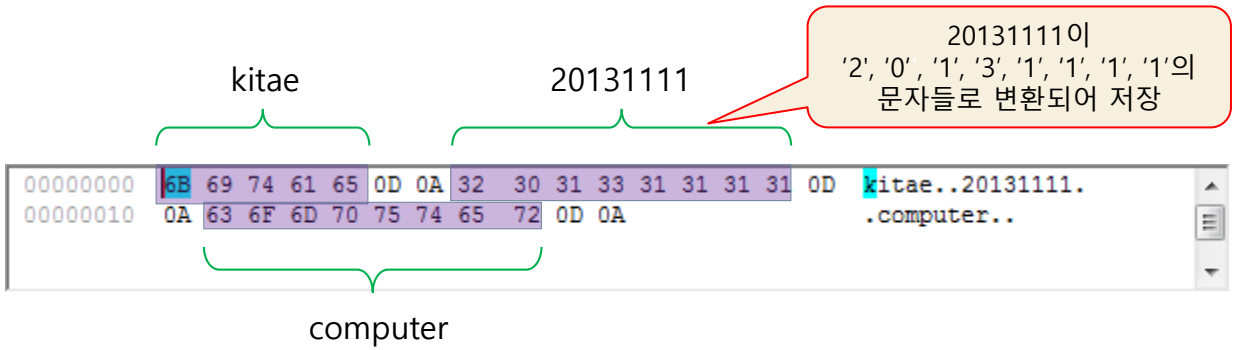
키보드로 입력 받아 텍스트 파일 저장하기


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

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

    // 키보드로부터 읽기
    cout << "이름>>";
    cin >> name
    cout << "학번>>";
    cin >> sid;
    cout << "학과>>";
    cin >> dept;

    // 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

    fout.close(); // 파일 닫기
}
```

정수 sid가 문자열로 변환되어 저장됨

```
이름>>kitae
학번>>20131111
학과>>computer
```





ifstream과 >> 연산자로 텍스트 파일 읽기


```
#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
  - 사례)
    - 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
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.

파일 모드 설정


```
void open(const char * filename, ios::openmode mode)
  mode로 지정된 파일 모드로 filename의 파일을 연다.
```

파일 모드 지정

- 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);
```

- 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
```

```
int get() //파일에서 한 문자를 읽어 리턴. 끝으면 EOF(-1) 반환

put(char ch) //파일에 한 문자 ch 기록
```

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 '₩r₩n'의 두 바이트를 '₩n'의 한 바이트로 리턴한다. c:₩windows₩system.ini는 총 13 라인의 219 바이트이지만, 실제 읽은 바이트 수는 각 라인의 '₩r' 개수 만큼 13개 모자란 206으로 카운트 된다.

get()으로 파일의 끝을 인지하는 방법


```
while(true) {
   int c = fin.get(); // 파일에서 문자(바이트)를 읽는다.
   if(c == EOF)     //마지막 읽은 EOF(-1) 값이 c에 리턴된다.

  {
      .....      // 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
}
```

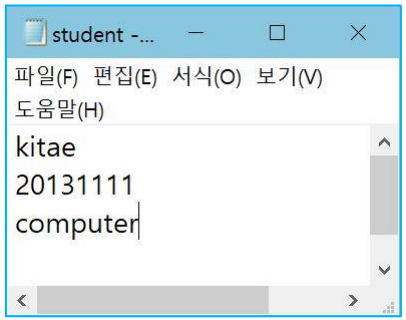
텍스트 파일 연결

fstream을 이용하여 c:₩temp₩student.txt 파일에 c:₩windows₩system.ini를 덧붙이는 프로그램을 작성하라.

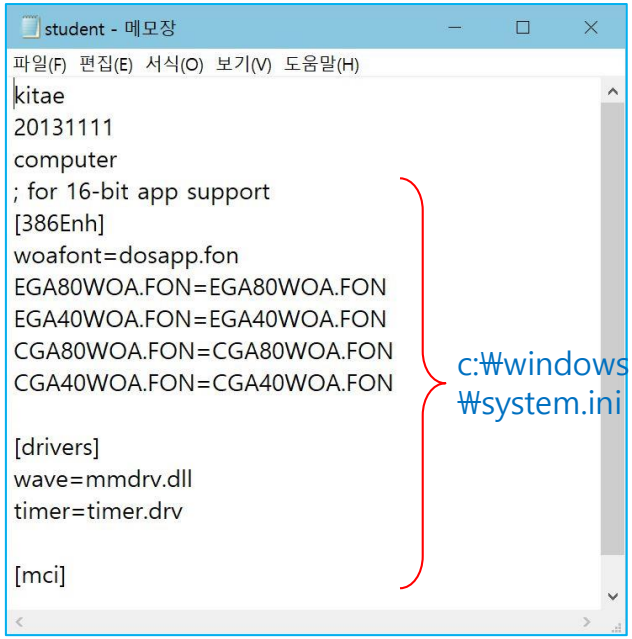
```
#include <iostream>
#include <fstream>
using namespace std;

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



원본 c:₩temp₩student.txt 파일



c:₩windows₩system.ini 본문

변경된 c:₩temp₩student.txt 파일

텍스트 파일의 라인 단위 읽기

- 두 가지 방법
  - istream의 getline(**char* line**, int n) 함수 이용
  - getline(ifstream& 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(ifstream& 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
}
```

istream의 getline()을 이용하여 텍스트 파일을 읽고 화면 출력

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 구성
   while(fin.getline(buf, 81)) // getline() 성공 True, 실패 False 반환
   {
      cout << buf << endl;      // 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]
```

getline(ifstream&, string&) 으로 라인읽기

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

int main()
{
ifstream fin("c:₩₩windows₩₩system.ini");
string line;

if (!fin)
{
  cout << "c:₩₩windows₩₩system.ini 열기 실패" << endl;
  return 0;
}

while (getline(fin, line)) // fin 파일에서 한 줄 읽기
{
     cout << line << endl;  // 라인 출력
}

fin.close();
}
```

String 클래스 문자열

전역변수 getline()

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::binary가 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
```

바이너리 I/O로 파일 복사

get(), put() 함수를 이용하여 c:₩temp에 있는 desert.jpg를 c:₩temp₩copydesert.jpg로 복사하라. 예제 실행 전에 desert.jpg를 미리 c:₩temp에 복사해두어야 한다.

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

int main() {
    // 소스 파일과 목적 파일의 이름
    const char* srcFile = "국화.jpg";
    const char* destFile =  " 국화Copy.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 파일 경로명

복사 대상 파일명

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 n개의 바이트를 배열 s에 읽어 들임. 파일의 끝을 만나면 읽기 중단
ostream& write(char* s, int n)
  배열 s에 있는 처음 n개의 바이트를 파일에 저장
int gcount()
  최근에 파일에서 읽은 바이트 수 리턴
```

read()로 텍스트 파일을 바이너리 I/O로 읽기

read()를 이용하여 한번에 100바이트씩 c:₩windows₩system.ini 파일을 읽어 화면에 출력하는 프로그램을 작성하라.

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

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[100];
    while(!fin.eof()) // eof() 이면 true 반환, 아니면 false
    {       fin.read(s, 100); // 최대 100 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

read()/write()로 이미지 파일 복사

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

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

int main() {
    const char* srcFile =  " 국화.jpg";
    const char* destFile =  " 국화Copy2.jpg";

    ifstream fsrc(srcFile, ios::in | ios::binary);
    if(!fsrc) {
        cout << srcFile << " 열기 오류" << endl;
        return 0;
    }
    ofstream fdest(destFile, ios::out | ios::binary);
    if(!fdest) {
        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:₩temp 폴더의 tulips.jpg의 경로명

c:₩temp₩copytulips.jpg로 복사

int 배열과 double 값을 바이너리 파일에 저장하고 읽기


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

int main() {
    char* file = "data.dat";

    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();

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

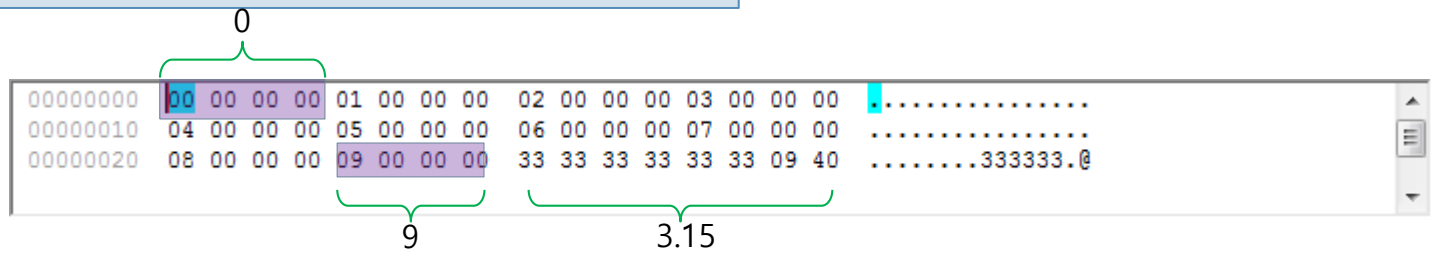
바이너리 I/O 모드 설정

write()로 한번에 배열을 쓴다.

```
    // 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;
    }

    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();
}
```

read()로 한번에 배열을 읽는다.

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

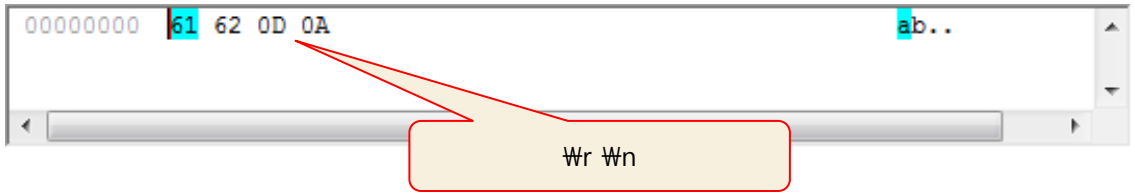


data.dat 파일 내부(바이너리 파일)

텍스트 I/O와 바이너리 I/O의 실행 결과 비교

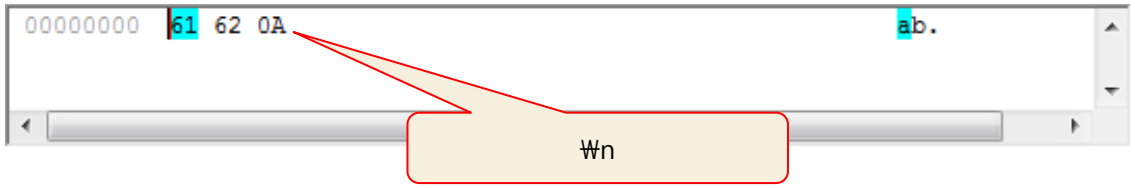
텍스트 I/O 모드

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



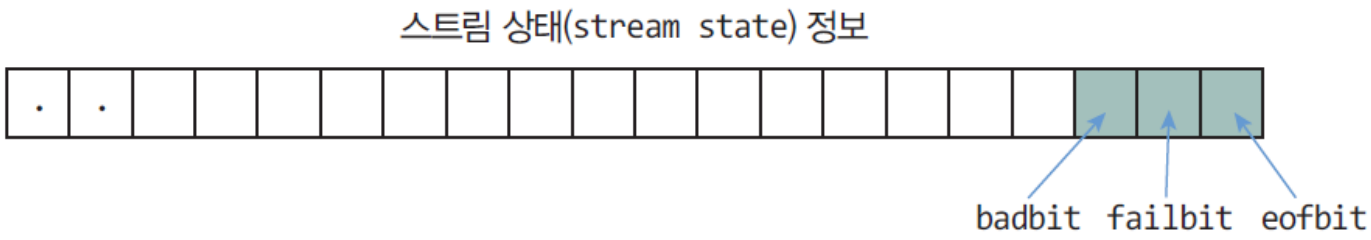
바이너리 I/O 모드

```
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
```



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 상태를 나타내는 비트 정보와 스트림 상태를 검사하는 멤버 함수

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