12.1 텍스트 파일과 바이너리 파일

파일 저장매체에 저장되는 정보

텍스트파일

바이너리파일

텍스트파일 글자와 문자만으로 구성되는 문서파일

ASCII코드 컴퓨터의 등장 초기 영어 문자 코드 체계

유니코드 각 글자에 2바이트 바이너리 코드 부여

텍스트파일 내부 보기

텍스트파일과 enter키 \r 커서를 현재 줄의 맨 앞으로 옮김

\n 커서를 현재 줄의 위치에서 한 줄 밑으로 옳기도록하는 지시

바이너리파일 이미지파일, 오디오파일,

바이너리 파일 내부 보기

hwp파일은 텍스트 파일인가? 바이너리 파일인가? 바이너리 파일

check time 1. 바이너리 파일

2. 1, 4, 7

12.2 파일 입출력 개요

C++ 파일 입출력 라이브러리 ifstream 파일읽기

ofstream 파일쓰기

fstream 하나의 파일에 읽고 쓰기

파일입출력 스트림은 파일을 프로그램과 연결한다 ifstream, ofstream 파일을 프로그램과 연결하는 스트림

get(), getline(), put(), read(), write() 함수들이 파일입출력에도 쓰임

헤더파일과 namespace <fstream>헤더파일, std이름공간 필요 🡪 #include <fstream>

using namespace std;

파일 입출력 모드: 텍스트I/O와 바이너리I/O 텍스트I/O 읽고쓰는 데이터를 문자로 국한

텍스트 파일을 읽고 쓸 때만 사용

바이너리I/O 바이너리 값 그대로 읽고 씀

모든 파일을 바이트의 스트림을 다룸 🡪 텍스트 파일, 바이너리 파일 상관 X

12.3 <<와 >>연산자를 이용한 간단한 텍스트 파일 입출력

파일쓰기를 위한 스트림객체 생성 파일출력스트림객체 생성 형식 ofstream 파일출력스트림객체이름

ex) ofstream fout

파일열기 파일연결 파일출력스트림객체.open() 파일출력 스트림객체의 open()멤버함수 호출

형식 파일출력스트림객체.open(“파일이름”)

ex) fout.open(“song,txt”)

생성자 생성자에 파일이름을 인수로 주어 파일연결

형식 ofstream 파일출력스트림객체(“파일이름”)

ex) ofstream fout(“song.txt”)

파일열기 성공 검사 !연산자 operator!() ios클래스의 멤버 🡪 상속받은 모든 입출력 클래스에서 사용가능

파일열기 성공 true

파일열기 실패 false

ex) if( !fout ) { …//파일열기가 실패한 경우 처리코드 }

파일출력스트림객체.is\_open() 파일열기 성공 true

파일열기 실패 false

ex) if( !fout.is\_open() ) { …//파일열기가 실패한 경우 처리코드 }

<<연산자를 이용한 파일 쓰기 <<연산자는 문자만 저장함

ex) int age = 21;

char singer[] = "kim";

char song[] = "Yesterday";

ofstream fout("song.txt");

if (!fout) return 0;

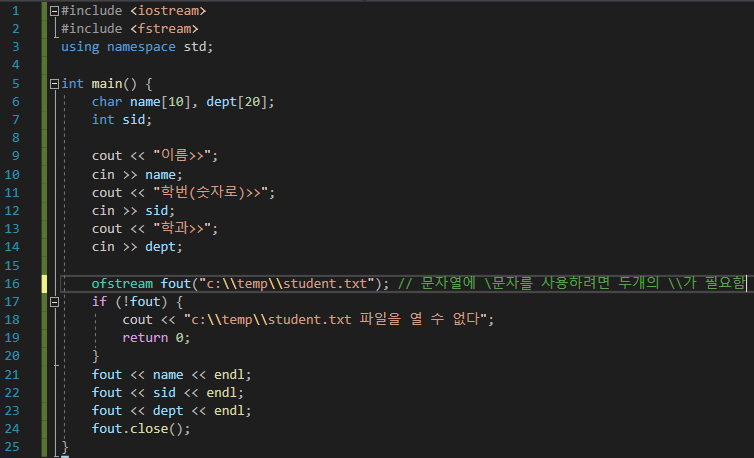
fout << age << "\n";

fout << singer << endl;

fout << song << endl;

파일닫기 파일연결끊기 형식 파일출력스트림객체.close()

ex) fout.close()

예제12-1 

>>연산자를 이용한 텍스트 파일 읽기 ifstream클래스를 이용해 파일입력스트림 객체 생성

파일입력스트림객체 생성 형식 ifstream 파일입력스트림객체이름

ex) ifstream fin

파일연결 파일입력스트림객체.open()

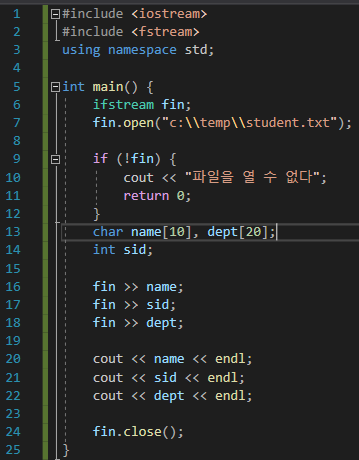
형식 파일입력스트림객체.open(“파일이름”)

ex) fin.open(“song,txt”)

생성자 생성자에 파일이름을 인수로 주어 파일연결

형식 ifstream 파일입력스트림객체(“파일이름”)

ex) ifstream fin(“song.txt”)

예제12-2 

check time ofstream fout

fout << tel << name

fout.close()

12.4 파일모드

파일모드란? 파일을 열 때 앞으로 어떤 파일 입출력을 수행할 것인지 알리는 정보

ios에 상수로 선언되어 있음

ios::in 읽기위해 파일을 연다

ios::out 쓰기위해 파일을 연다

ios::ate 쓰기 위해 파일을 염, 열기 후 파일포인터를 파일 끝에 둠

파일포인터를 옮겨 임의의 위치에 쓸수 있음

ios::app 파일 쓰기시에만 적용

파일쓰기 시마다 파일포인터를 맨 끝으로 옮겨놈 🡪 파일 끝에서만 파일의 끝에 쓰기가 이루어짐

ios::trunc 파일을 열 때, 파일의 내용을 모두 지움

ios::binary 바이너리 IO로 파일을 염

파일모드설정 open함수, 생성자로 파일모드 설정

open() 형식 파일객체.open(“파일이름”, 파일모드)

|를 사용해 여러 개의 파일모드 동시 지정 가능

ex) fout.open(“student.txt”, ios::out | ios::app | ios::binary);

ex) ifstream fin; // student.txt 처음부터 읽고자 하는 경우

fin.open(“student.txt”);

ex) ofstream fout; // student.txt 파일의 끝에 데이터를 저장하는 경우

fout.open(“student.txt”, ios::out | ios::app)

ex) fstream fbinout; // 바이너리IO로 data.bin 파일을 기록하는 경우

fbinout.open(“data.bin”, ios::out | ios::binary)

스트림객체 생성자 형식 스트림객체(“파일이름”, 파일모드)

ex) ifstream fin(“student.txt”);

ofstream fout(“student.txt”, ios::out | ios::app);

fstream fbinout(“data.bin”, ios::out | ios::binary);

12.5 멤버함수를 이용한 텍스트I/O

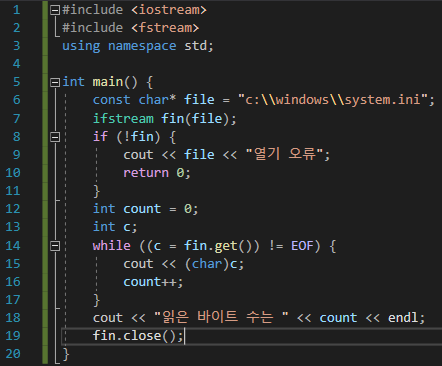
get(), put함수 이용

텍스트 I/O모드로 열기 ios::binary 지정 X 🡪 텍스트I/O

get()과 put()을 이용한 텍스트I/O int get() 문자 1개(1바이트)를 읽어 리턴

파일의 끝에서 읽으면 EOF리턴

ostream& put(char ch) 파일에 문자 ch 기록

예제12-3 

get()과 EOF get() 한 바이트를 읽어 파일포인터를 다음 위치로 전진

마지막 문자를 읽으면, 파일의 끝(EOF)을 가리킴(가리키기만 하고 읽지 않은 상태 🡪 eof()가 false리턴)

EOF를 get()이 읽으면 eof를 반환∙eofbit플래그를 1로 설정 🡪 eof가 true리턴

get()이 리턴한 값과 EOF를 비교 while (true) {

int c = fin.get(); //바이트 읽기

if (c == EOF) { //파일의 끝을 만난 경우

...

break;

}

else {} //읽은 문자 c 처리

}

or

while ((c = fin.get()) != EOF) {

...//읽은 문자 처리

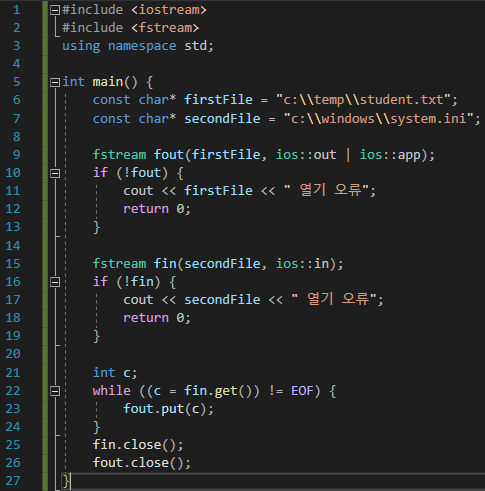
}

파일의 끝을 잘못 인지하는 코드 while (!fin.eof()) { // 전 루프 때, 마지막 바이트를 읽고, 파일포인터가 eof를 가리킴 --> eof 통과

int c = fin.get(); //eof를 읽음 --> -1 리턴

}

ios::app모드로 텍스트 파일 덧붙여쓰기 파일에 끝에 다른 파일 덧붙여 쓰기

예제12-4 

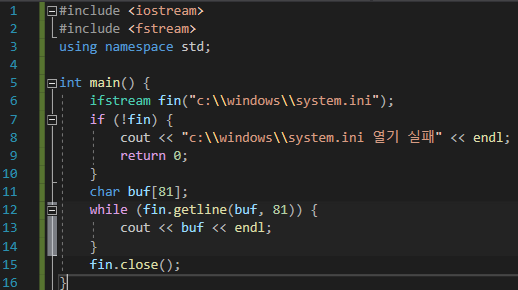
텍스트 파일의 라인 단위 읽기 istream getline(char\* line, int n) 한줄을 읽는 함수

char buf[81];

ifstream fin("c:\\windows\\system.ini");

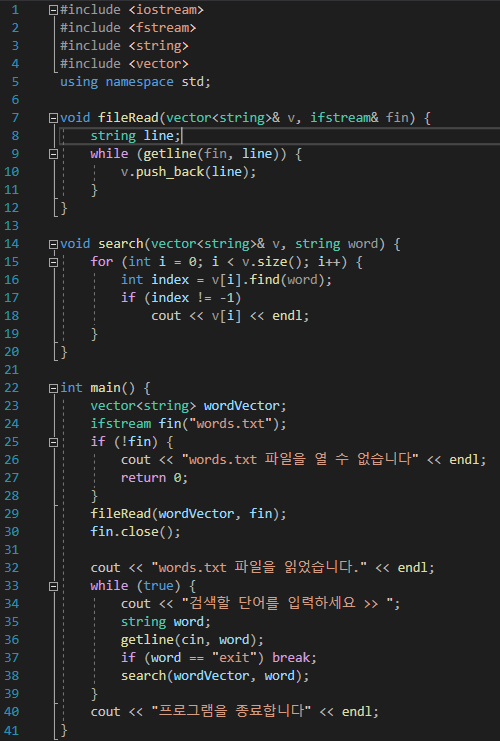
while (fin.getline(buf, 81)) {}//한라인이 최대 80개의 문자로 구성, 끝에 \0추가

istream ()연산자 현재 스트림 내에 오류상태를 검사하고 오류가 발생하면 false를 리턴함

예제12-5 

ifstream& getline\*(ifstream& fin, string& line) string으로 문자열을 받음

<string>헤더파일에 선언됨

예제12-6 

12.6 바이너리I/O

파일의 각 바이트를 바이너리값 그대로 읽음

변수나 버퍼의 바이너리 값을 그대로 파일에 저장 🡪 저수준의 입출력 방식

바이너리I/O로 파일 열기 파일을 열 때 ios::binary 파일 모드 지정

ex) ifstream fin;

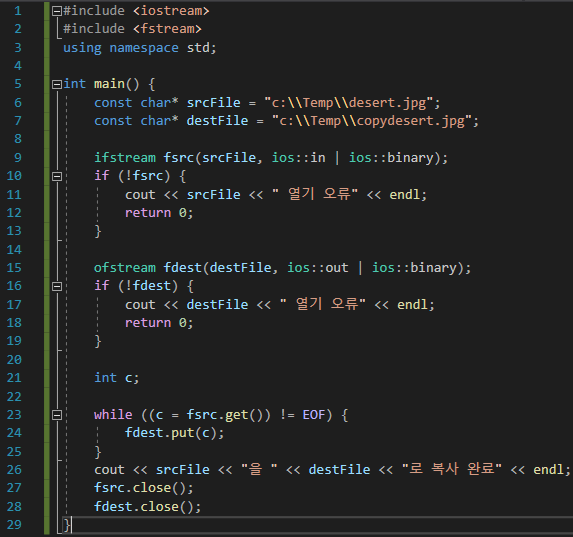
fin.open("desert.jpg", ios::in | ios::binary);

ofstream fout("desert.jpg", ios::out | ios::binary);

fstream fsin("desert.jpg", ios::in | ios::binary);

get()와 put()을 이용한 바이너리I/O get()과 put() 바이너리 바이트를 읽고 쓰는 데도 사용

사용법 동일

예제12-5 

read()/write()로 블록단위 파일 입출력 블록 단위로 입출력 🡪 get(), put()보다 실행속도가 빠름

read() istream의 멤버

ifstream이 상속받음

함수원형 istream& read(char\* s, int n)

파일에서 최대 n개의 바이트를 읽음, 파일의 끝을 만나면 읽기 중단

EOF를 읽으면 읽기를 중단하기 때문에 n개의 바이트를 읽었다고 보장 불가 🡪 gcoun()로 실제 일은 바이트 수 변환

write() ostream의 멤버

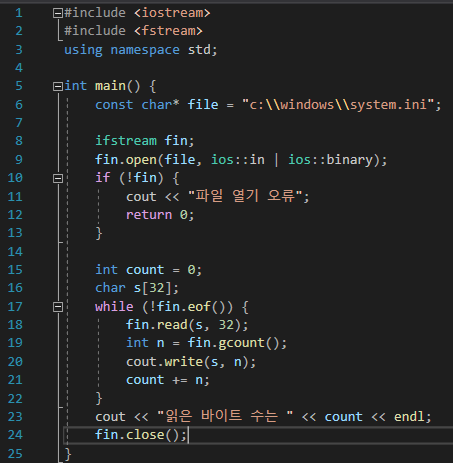
ofstream이 상속받음

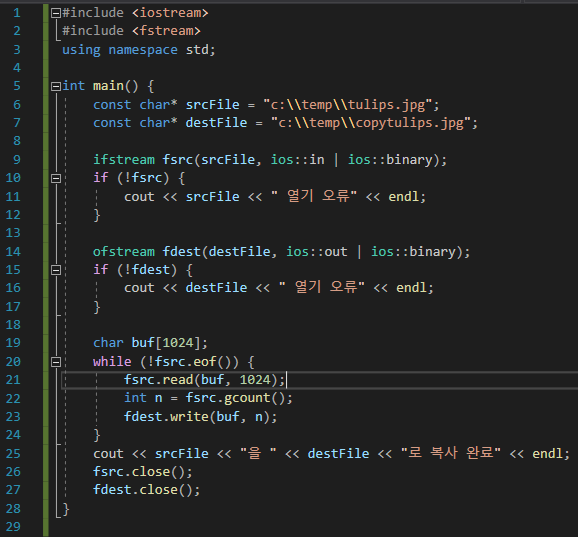
함수원형 ostream& write(char\* s, int n)

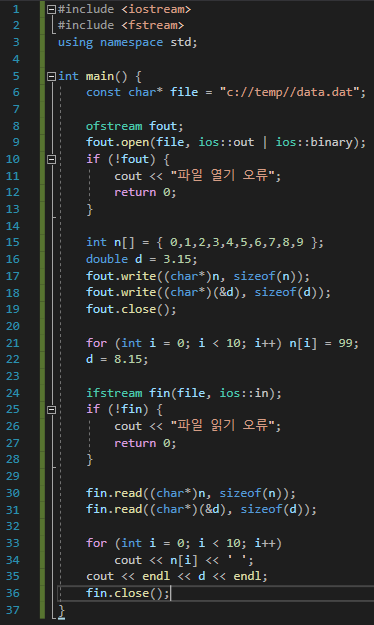
배열 s에 있는 처음 n개의 바이트를 파일에 저장

gcount() 함수원형 int gcount()

최근에 파일에서 읽은 바이트 수 리턴

예제12-8 

예제12-9 

예제12-10 

바이너리I/O와 텍스트I/O의 확실한 차이점 파일의 끝을 인식하는 방법에는 차이가 없음 파일모드와 관계없이 EOF와 비교

텍스트I/O와 바이너리I/O는 ‘\n’을 읽고쓸 때 서로 다르게 작동함 텍스트I/O \n을 쓰면 \r\n이 쓰임

\r\n을 읽으면 \n이 읽힘

비이너리I/O 그대로 읽고 쓰임

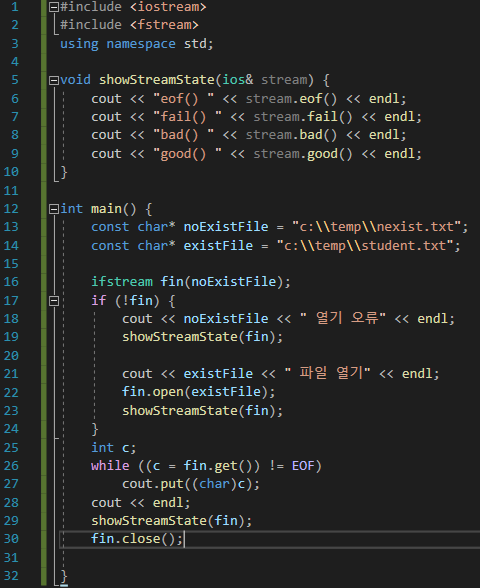
<<,>>연산자는 텍스트I/O에서만 사용가능

12.7 스트림 상태 검사

스트림의 상태를 저장하는 맴버 변수 입출력이 진행되는 동안 발생한 오류 정보 유지

오류 비트를 이용해 정보를 알림 ex) eofbit, failbit, badbit…

스트림의 상태를 검사하는 멤버함수 ex)eof(), fail(), bad(, good(), clear()

예제12-11 

EOF를 나타내는 데이터 파일의 끝을 인식하는 작업 = os의 몫

파일의 끝을 알려주는 것은 os의 역할

12.8 임의접근

임의접근 파일 내의 임의의 위치로 옮겨 다니면서 읽고 씀

파일포인터 파일내에 읽고 쓸 바이트의 위치를 가리키는 포인터

get pointer 파일 내의 읽기 지점을 가리키는 파일 포인터

put pointer 파일 내의 쓰기 지점을 가리키는 파일 포인터

ifstream get포인터만을 가짐

get(), read(), getline(), >>등의 읽기 연산은 get포인터에서 읽고, 바이트 수 만큼 get포인터 전진

ofstream put포인터만을 가짐

put(), write(), <<등의 쓰기 연산은 put포인터에서 쓰고, 바이트 수 만큼 put포인터 전진

파일모드와 파일포인터 파일포인터의 위치는 파일모드에 따라 다르게 설정됨

ios::in get pointer 파일의 처음을 가리킴

ios::out put pointer 파일의 처음을 가리킴

ios::ate put pointer 파일의 끝을 가리킴

ios::app 열 때의 put pointer 파일의 처음

쓸 때의 put poiner 파일의 끝

임의접근방법 파일포인터의 위치를 임의의 위치로 이동

파일포인터 멤버함수 istream& seekg(streampos pos) 정수값으로 주어진 절대위치 pos로 get pointer를 옮김

istream& seekg(streamoff offset, ios::seekdir seekbase) seekbase를 기준으로 offset만큼 떨어진 위치로 get pointer를 옮김

ostream& seekp(streampos pos) 정수값으로 주어진 절대 위치 pos로 put pointer를 옮김

ostream& seekp(streamoff offset, ios::seekdir seekbase) seekbase를 기준으로 offset만큼 떨어진 위치로 put pointer를 옮김

streampos tellg() 입력스트림의 현재 get pointer값 리턴

streampos tellp() 입력스트림의 현재 put pointer값 리턴

seekbase에 사용되는 상수 ios::beg 파일의 처음 위치 기준

ios::cur 현재 파일 포인터 기준

ios::end 파일의 끝 위치 기준

파일포인터는 인덱스 0부터 시작함

절대 위치로 파일포인터 옮기기 istream& seekg(streampos pos)이용

ex) fin.seekg(5);

int n = fin.tellg();

int c = fin.get();

상대 위치로 파일포인터 옮기기 seekbase를 기준으로 offset만큼 옮기기

ex) fin.seekg(10, ios::beg); //파일시작점 기준으로 getpointer를 10바이트 전진

fin.seekg(-1, ios::cur); //현재 위치에서 getpointer를 1바이트 후진

fin.seekg(0, ios::end); //파일의 맨끝 위리로 get pointer를 옮김

//여기서부터 읽을 데이터는 없다

ex) /\*파일의 맨 마지막 문자 읽기\*/

fin.seekg(-1, ios::end); //파일의 마지막 문자 위치로 getpointer를 보냄

int c = fin.get(); //파일의 마지막 문자를 읽음

ex) /\*10바이트 간격으로 문자읽기\*/

int c;

while ((c = fin.get() != EOF) { //get()으로 1바이트 읽으면서 전진

fin.seekg(9, ios::cur); //9바이트 전진

}

ex) /\*텍스트 파일을 거꾸로 화면에 출력하는 경우\*/

fin.seekg(0, ios::end); //EOF위치로 get pointer이동

int fileSize = fin.tellg(); //get ponter값 = 파일의 크기

for (int i = 0; i < fileSize; i++) {

fin.seekg(fileSize - 1 - i, ios::beg); //파일의 처음을 기준으로 get pointer이동

int c = fin.get();

cout << (char)c;

}

파일포인터를 이용하여 파일 크기 알아내기 파일포인터를 파일의 맨 끝으로 이동시킨 파일포인터의 위치 = 파일크기

예제12-12 