제15장

파일 처리

Perfect C

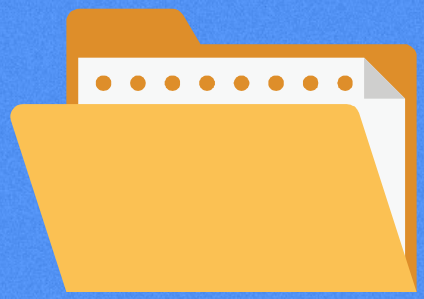
C언어로 배우는 프로그래밍 기초

단원 목표

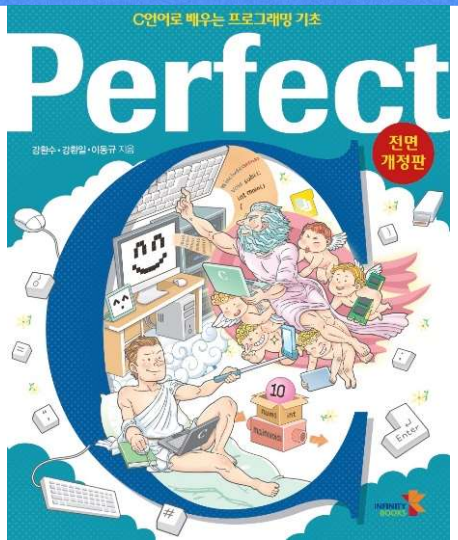
학습목표

- **텍스트 파일과 이진 파일의 차이를 이해하고 설명할 수 있다.**
  - 주기억장치와 파일의 차이
  - 텍스트 파일과 이진 파일의 정의와 예
  - 파일 스트림과 파일모드의 이해
  - 함수 fopen()과 fopen_s(), fclose()의 사용
- **텍스트 파일의 입출력 함수를 이해하고 설명할 수 있다.**
  - 함수 fprintf()와 fscanf(), fscanf_s()의 사용
  - 함수 fputc()와 fgetc()의 사용
- **이진 파일의 입출력 함수를 이해하고 설명할 수 있다.**
  - 함수 fwrite()와 fread()의 사용
  - 함수 getw()와 putw()의 사용
- **파일 삭제 및 이름 바꾸기 함수를 이해하고 설명할 수 있다.**
  - 함수 remove()와 rename()의 사용

학습목차



01. 파일 기초



파일의 필요성

- **워드프로세서 없이 프로그램에서 프로그램의 결과로 구성되는 파일을 직접 만들 수 있을까?**
  - 프로그램에서 출력을 파일에 한다면 파일이 생성
  - 반대로 키보드에서 표준 입력하던 입력을 파일에서 입력하면 파일 입력

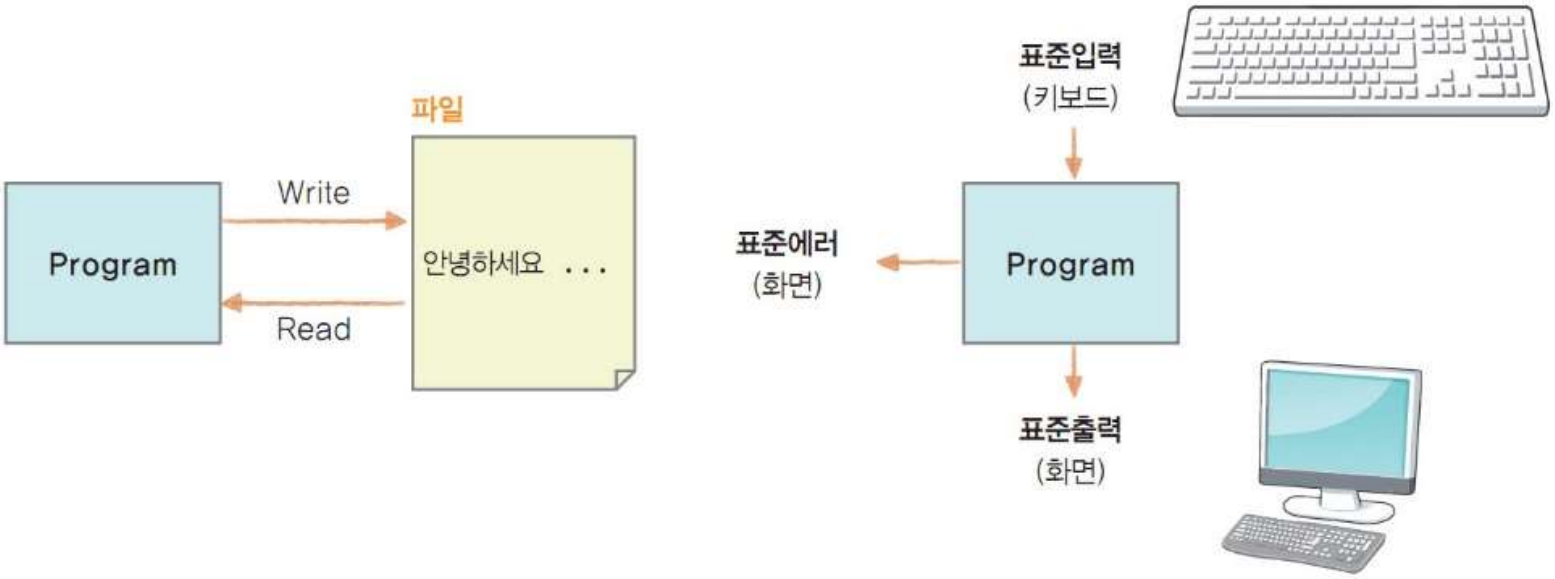


그림 15-1 파일입출력과 표준입출력

파일과 메모리

- **프로그램이 종료되면 모두 사라지는 자료**
  - 변수와 같이 프로그램에서 내부에서 할당되어 사용되는 주기억장치의 메모리 공간
- **프로그램이 종료되더라도 계속 저장**
  - 보조기억장치인 디스크에 저장되는 파일(file)은 직접 삭제하지 않은 한 계속 남음
- **프로그램에서 사용하던 정보를 종료 후에도 계속 사용하고 싶다면**
  - 프로그램에서 파일에 그 내용을 저장
  - 학생 성적 처리를 프로그램을 통하여 결과를 얻어내 그 처리 결과를 지속적으로 저장하려면 파일에 저장

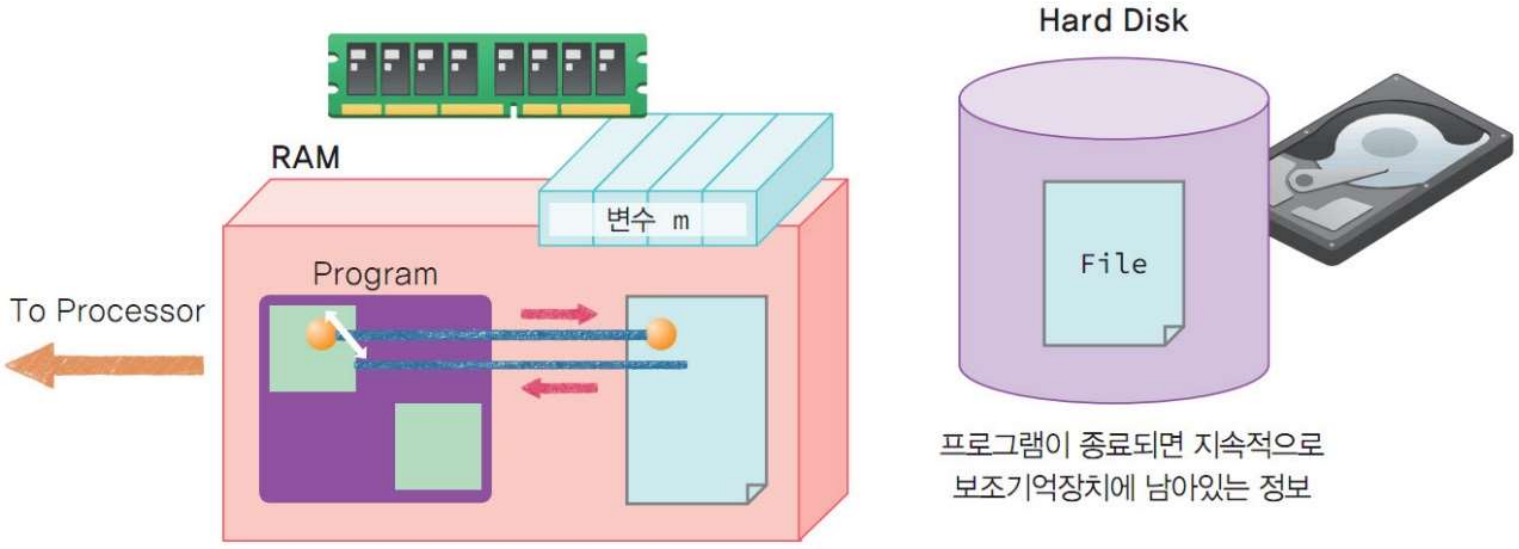


그림 15-2 주기억장치의 저장공간과 파일의 차이

텍스트 파일과 이진파일

- **파일: 보조기억장치의 정보저장 단위로 자료의 집합**
  - 텍스트 파일(text file)과 이진 파일(binary file) 두 가지 유형으로 나뉨
- **텍스트 파일: 메모장(notepad) 같은 편집기로 작성된 파일**
  - 내용이 아스키 코드(ascii code)와 같은 문자 코드값으로 저장
  - 메모리에 저장된 실수와 정수와 같은 내용도 문자 형식으로 변환되어 저장
  - 텍스트 편집기를 통하여 그 내용을 볼 수 있고 수정 가능
- **이진 파일: 실행파일과 그림 파일, 음악 파일, 동영상 파일 등**
  - 목적에 알맞은 자료가 이진 형태(binary format)로 저장되는 파일
  - 자료는 메모리 자료 내용에서 어떤 변환도 거치지 않고 그대로 파일에 기록
  - 입출력 속도도 텍스트 파일보다 빠름
  - 메모장과 같은 텍스트 편집기로는 그 내용을 볼 수 없음
  - 내용을 이미 알고 있는 특정한 프로그램에 의해 인지될 때 의미가 있음

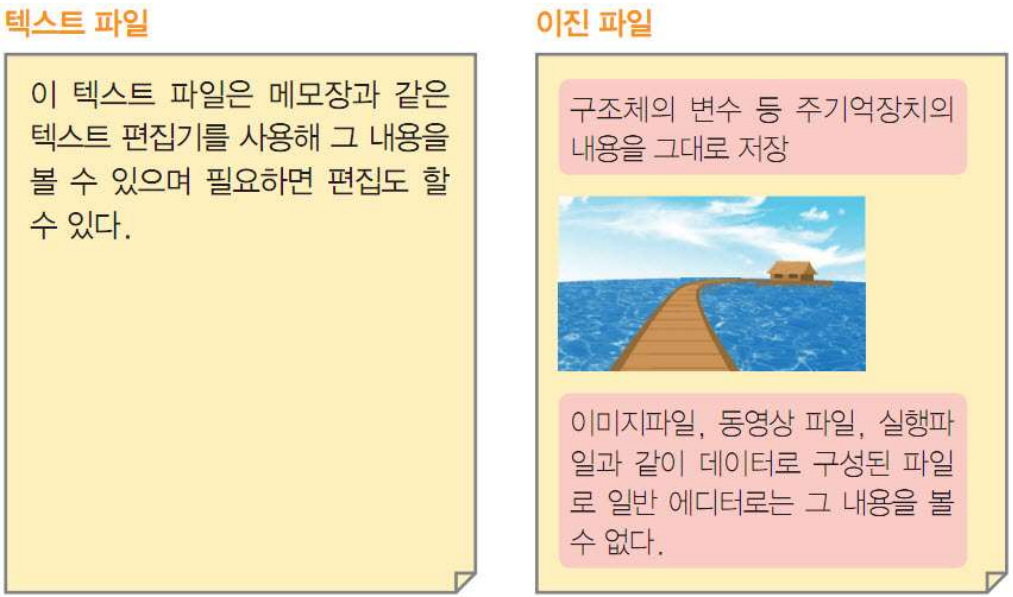


그림 15-3 텍스트 파일과 이진 파일

입출력 스트림(1)

- **자료의 입력과 출력은 자료의 이동**
  - 자료가 이동하려면 이동 경로가 필요
- **입출력 스트림(io stream)**
  - 입출력 시 이동 통로
- **표준입력 스트림: 키보드에서 프로그램으로 자료가 이동 경로**
  - 함수 scanf()
    - 표준 입력 스트림에서 자료를 읽을 수 있는 함수
- **표준출력 스트림: 프로그램에서 모니터의 콘솔로 자료가 이동 경로**
  - 함수 printf()
    - 표준출력 스트림으로 자료를 보낼 수 있는 함수

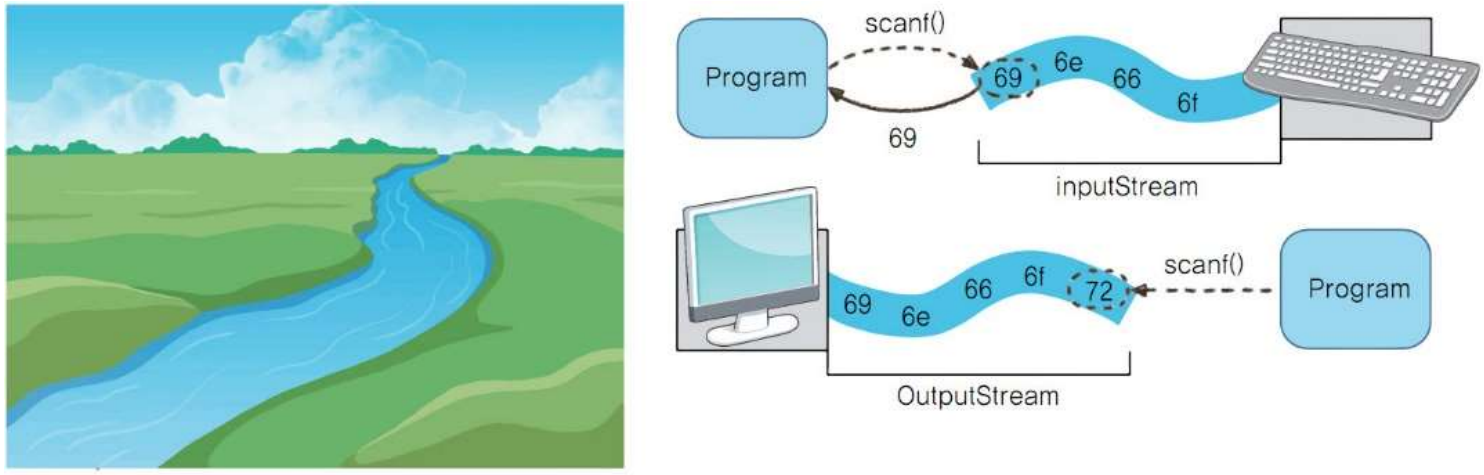


그림 15-4 스트림의 이해

입출력 스트림(2)

- **입력 스트림(input stream): 다른 곳에서 프로그램으로 들어오는 경로**
  - 자료가 떠나는 시작 부분이 자료 원천부(data source)
  - 표준입력: 원천부가 키보드
    - 파일입력: 파일이면 파일로부터 자료를 읽는 것
    - 스크린입력: 터치스크린이면 스크린에서 터치 정보
    - 네트워크입력: 다른 곳에서 프로그램으로 네트워크를 통해 자료가 전달
- **출력 스트림(input stream) : 프로그램에서 다른 곳으로 나가는 경로**
  - 자료의 도착 장소가 자료 목적부(data destination)
  - 표준출력: 목적부가 콘솔
    - 파일출력: 파일이면 파일에 원하는 값을 저장
    - 프린터출력: 프린터이면 프린터에 출력물
    - 네트워크출력: 네트워크이면 네트워크 출력이 되어 다른 곳으로 자료가 이동

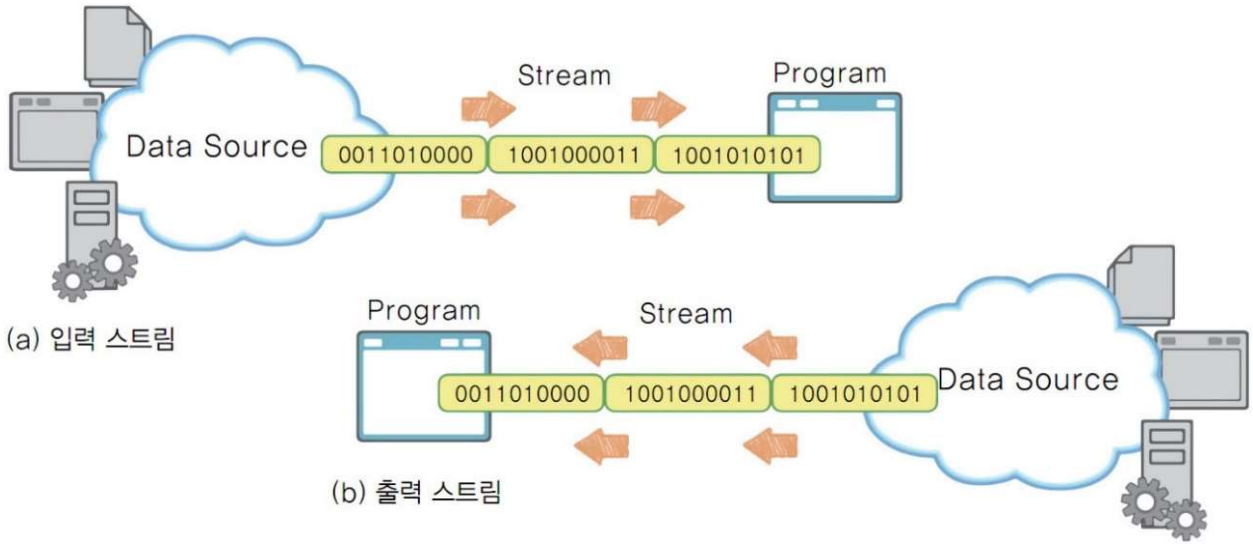


그림 15-5 입력 스트림과 출력 스트림

파일 스트림 이해

- **파일 스트림**
  - 보조기억장치의 파일과 프로그램을 연결하는 전송경로
  - 파일 입력 스트림(file input stream)
    - 파일에서 프로그램으로 자료의 입력을 위한 스트림
  - 마찬가지로 파일 출력 스트림(file output stream)
    - 프로그램에서 파일로 출력을 위한 스트림
- **파일 스트림을 만들기 위해서는**
  - 특정한 파일이름과 파일모드가 필요
    - 여기서 파일 모드란 입력 또는 출력과 같은 스트림의 특성

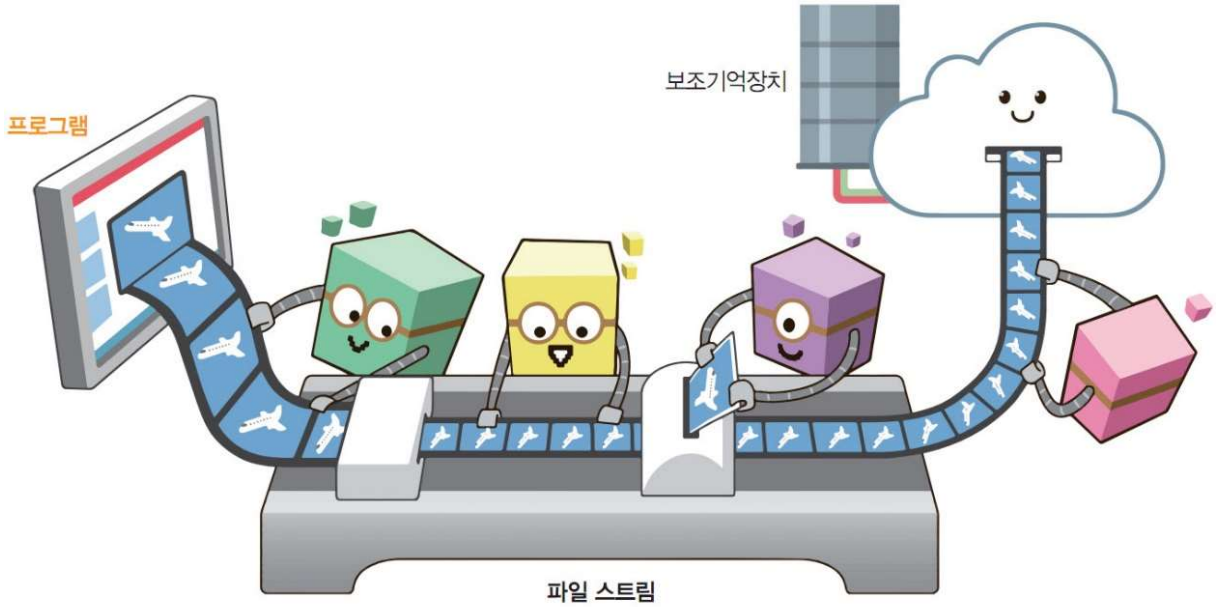


그림 15-6 파일 스트림 개념

함수 fopen(): 파일 스트림 열기

- **함수 fopen() 또는 fopen_s()를 이용**
  - 프로그램에서 특정한 파일과 파일 스트림을 연결하는 함수
    - 헤더 파일 stdio.h 필요
  - 현재 Visual C++에서 함수 fopen()는 fopen_s()로 대체, 함께 사용 가능
- **FILE: 헤더 파일 stdio.h에 정의되어 있는 구조체 유형**
  - 함수 fopen()의 반환값 유형 FILE *은 구조체 FILE의 포인터 유형
  - 함수 fopen()은 인자가 파일이름과 파일열기 모드
    - 파일 스트림 연결에 성공하면 파일 포인터를 반환하며, 실패하면 NULL을 반환

```
struct _iobuf {
        char *_ptr;
        int   _cnt;
        char *_base;
        int   _flag;
        int   _file;
        int   _charbuf;
        int   _bufsiz;
        char *_tmpfname;
        };
typedef struct _iobuf FILE;
```

그림 15-8 구조체

```
if ( (f = fopen(fname, "w")) == NULL )
{
   printf( "파일이 열리지 않습니다.\n" );
   exit(1);
};
```

함수 fopen()과 fopen_s() 함수원형

```
FILE * fopen(const char * _Filename, const char * _Mode);
errno_t fopen_s(FILE ** _File, const char * _Filename, const char * _Mode);
```

- 함수 fopen()은 파일명 _Filename의 파일 스트림을 모드 _Mode로 연결하는 함수이며, 스트림 연결에 성공하면 파일 포인터를 반환하며, 실패하면 NULL을 반환한다.
- 함수 fopen_s()는 스트림 연결에 성공하면 첫 번째 인자인 _ File에 파일 포인터가 저장되고 정수 0을 반환하며, 실패하면 양수를 반환한다. 현재 Visual C++에서는 함수 fopen_s()의 사용을 권장하고 있다.

그림 15-7 함수 fopen()과 fopen_s() 함수원형

함수 fopen_s()

- **파일 "basic.txt"를 여는 모듈**
  - 파일에 자료를 쓰기 위한 파일 스트림을 연결하기 위해서는 모드 값을 "w"로 기술
  - 함수 fopen_s()는 성공적으로 지정된 외부 파일 이름과 내부 파일 포인터 f와 출력 파일 스트림이 연결되면 FILE 포인터가 인자 f에 저장
    - 파일 스트림 연결에 성공하면 정수 0을 반환, 연결에 실패하면 양수를 반환
    - 첫 번째 인자는 파일 포인터의 주소값
    - 두 번째 인자인 문자열은 처리하려는 파일 이름
    - 세 번째 문자열은 파일열기 종류인 모드
- **파일열기 종류(모드)**
  - 텍스트 파일인 경우 "r", "w", "a" 등의 종류
  - 읽기모드 r
    - 읽기가 가능한 모드이며, 쓰기는 불가능
  - 쓰기모드 w
    - 파일 어디에든 쓰기가 가능한 모드 이나 읽기는 불가능
  - 추가모드 a
    - 파일 중간에 쓸 수 없으며
    - 파일 마지막에 추가적으로 쓰는 것만 가능한 모드
    - 읽기는 불가능
    - 파일에 쓰는 내용은 무조건 파일 마지막에 추가

```
if ( fopen_s(&f, "basic.txt", "w") != 0 )
//if ( (f = fopen(fname, "w")) == NULL )
{
    printf( "파일이 열리지 않습니다.\n" );
    exit(1);
};
...
fclose(f);
```

그림 15-9 함수 fopen_s()의 사용

함수 fclose()로 파일 스트림 닫기

- **함수 fclose()**
  - fopen()으로 연결한 파일 스트림을 닫는 기능을 수행
  - 파일 스트림을 연결한 후 파일 처리가 모두 끝났으면 파일 포인터 f를 인자로 함수 fclose()를 호출하여 반드시 파일을 닫도록
  - 내부적으로 파일 스트림 연결에 할당된 자원을 반납하고, 파일과 메모리 사이에 있던 버퍼의 내용을 모두 지우는 역할을 수행
  - 파일 스트림 f를 닫는 함수로서, 성공하면 0을 실패하면 EOF을 반환

함수 fclose()

```
int fclose(FILE * _File);
```

함수 fclose()는 파일 스트림 f를 닫는 함수로서, 성공하면 0을 실패하면 EOF을 반환한다.

```
fclose(f);
```

그림 15-10 함수 fclose() 함수원형

성적 파일 생성

예제 fopen.c

- 함수 fopen_s()와 fprintf()를 이용하여 지정한 학생이름과 점수를 파일 "basic.txt"에 출력하는 프로그램

fpritf()와 exit()

- 함수 fprintf()는 파일에 서식화된 문자열을 출력하는 함수
- 파일이 쓰기 모드로 열리지 않으면 함수 exit()를 이용하여 프로그램을 종료
  - 함수 exit()를 이용하려면 헤더 파일 stdlib.h가 필요
  - 함수를 강제로 종료하는 기능을 수행
  - 인자 값 0은 정상적인 종료를 의미하고, 0이 아닌 값은 정상적인 종료가 아님을 운영체제에게 알리는 의미로 사용

이름이 강미정인 학생의 성적은 99 입니다.

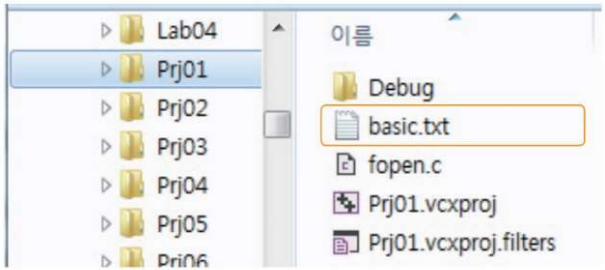


그림 15-11 예제 fopen 결과로 생성된 basic.txt 파일

실습예제 15-1 fopen.c

이름과 성적 정보를 입력하여 간단한 파일을 생성

```
01  // file: fopen.c
02  #define _CRT_SECURE_NO_WARNINGS
03  #include <stdio.h>
04  #include <stdlib.h> //for exit()
05
06  int main()
07  {
08     char *fname = "basic.txt"; //파일이름
09     FILE *f; //파일 포인터
10
11     //파일에 쓰려는 자료
12     char name[30] = "강미정";
13     int point = 99;
14
15     //파일 열기 함수 fopen()과 fopen_s()
16     if ( (f = fopen(fname, "w")) == NULL )
17     //if (fopen_s(&f, fname, "w") != 0)
18     {
19        printf("파일이 열리지 않습니다.\n");
20        exit(1);
21     };
22
23     //파일"basic.txt"에쓰기
24     fprintf(f, "이름이 %s인 학생의 성적은 %d 입니다.\n", name, point);
25     fclose(f);
26     //표준출력 콘솔에쓰기
27     printf("이름이 %s인 학생의 성적은 %d 입니다.\n", name, point);
28     puts("프로젝트 폴더에서 파일 basic.txt를 메모장으로 열어 보세요.");
29
30     return 0;
31  }
```

표준출력이 아니라 지정한 파일 f에 출력 printf()를 하는 기능을 수행한다.

표준출력에 출력을 수행한다.

설명

```
08    fname에 생성할 파일이름을 저장
09    파일포인터를 하나 선언
12~13 파일에 쓸 자료값을 저장
16    함수 fopen()의 호출 시, 첫 인자는 파일이름, 두 번째 인자는 모드로 "w"는 쓰기모드이며, 반환
      값을 파일 포인터로 선언한 f에 대입, 만일 파일 열기에 실패하면 f에 NULL이 저장됨
19~20 파일 열기에 실패하면 메시지 출력하고 exit(1)으로 종료
24    파일 basic.txt에 쓰기 위해 함수 fprintf() 사용, 첫 인자는 파일 포인터 f이며, 나머지는
      printf()와 동일하므로 파일에 "이름이 강미정인 학생의 성적은 99 입니다."를 쓰는 기능 수행
25    파일 처리가 종료되었으면 fclose()로 스트림을 닫음
26    표준출력에도 출력해 봄
```

실행결과

```
이름이 강미정인 학생의 성적은 99 입니다.
프로젝트 폴더에서 파일 basic.txt를 메모장으로 열어 보세요.
```

LAB 파일 “myinfo.txt”을 열어 간단한 정보를 출력

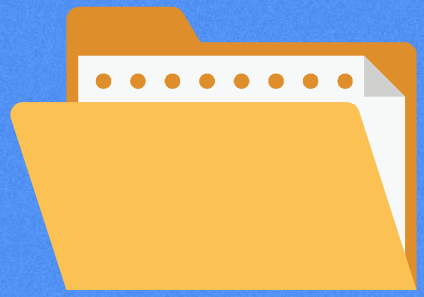
- **파일 “myinfo.txt”를 쓰기모드로 열어**
  - 전화번호, 주소, 나이 등의 간단한 정보를 출력
- **함수 fprintf(f, ...)**
  - 첫 인자에 파일 포인터를 대입하면 파일에 서식화된 문자열을 출력
- **파일 f를 닫기**
  - 파일에 출력이나 입력을 모두 마치면 함수 fclose(f)
- **결과**
  - 전화번호: 010-3018-9917, 주소:서초구 대치로 332, 나이: 22
  - 프로젝트 폴더에서 파일 myinfo.txt를 메모장으로 열어 보세요.

Lab 15-1 basicfileio.c

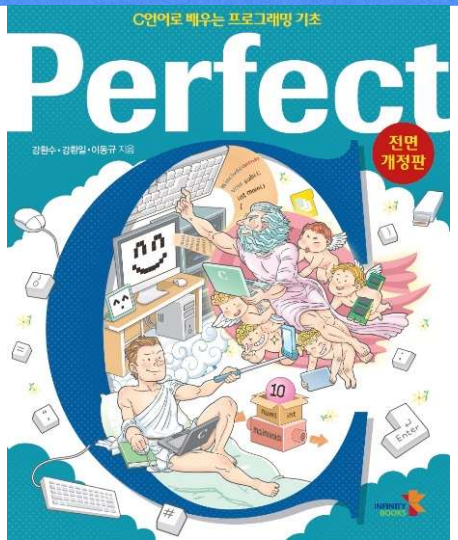
```
01  // file: basicfileio.c
02  #define _CRT_SECURE_NO_WARNINGS
03  #include <stdio.h>
04  #include <stdlib.h> //for exit()
05
06  int main()
07  {
08     FILE *f; //파일 포인터
09
10     //파일 열기 함수 fopen()과 fopen_s()
11     if ((f = fopen(____________________________)) == NULL)
12     //if (fopen_s(&f, "myinfo.txt", "w") != 0)
13     {
14        printf("파일이 열리지 않습니다.\n");
15        exit(1);
16     };
17
18     //파일에 쓰려는 자료
19     char tel[15] = "010-3018-9917";
20     char add[30] = "서초구 대치로 332";
21     int age = 22;
22     //파일"basic.txt"에 쓰기
23     fprintf(____, "전화번호: %s, 주소:%s, 나이: %d\n", tel, add, age);
24
25     //파일 닫기
26     ___________________;
27     //표준출력 콘솔에 쓰기
28     printf("전화번호: %s, 주소:%s, 나이: %d\n", tel, add, age);
29     puts("프로젝트 폴더에서 파일 myinfo.txt를 메모장으로 열어 보세요.");
30
31     return 0;
32  }
```

정답

```
11  if ((f = fopen("myinfo.txt", "w")) == NULL)
23  fprintf(f, "전화번호: %s, 주소:%s, 나이: %d\n", tel, add, age);
26  fclose(f);
```



02. 텍스트 파일 입출력



함수 fprintf()와 fscanf()

- **함수 fprintf()와 fscanf() 또는 fscanf_s()를 이용**
  - 텍스트 파일에 자료를 쓰거나 읽기 위하여
  - 헤더 파일 stdio.h를 포함
    - 첫 번째 인자는 입출력에 이용될 파일
    - 두 번째 인자는 입출력에 이용되는 제어 문자열
    - 다음 인자들은 입출력될 변수 또는 상수 목록
  - 함수 fprintf()와 fscanf() 또는 fscanf_s()의 첫 번째 인자에 각각 stdin 또는 stdout를 이용하면 표준 입력, 표준 출력으로 이용이 가능
- **기호 상수 stdin, stdout은 stderr**
  - 헤더 파일 stdio.h에 정의되어 있는 값
  - 각각 표준 입력, 표준 출력, 표준에러를 의미

함수 fprintf()와 fscanf() 함수원형

```
int fprintf(FILE * _File, const char * _Format, ...);
int fscanf(FILE * _File, const char * _Format, ...);
int fscanf_s(FILE * _File, const char * _Format, ...);
```

위 함수에서 _File은 서식화된 입출력 스트림의 목적지인 파일이며, _Format은 입출력 제어 문자열이며, 이후 기술되는 인자는 여러 개의 출력될 변수 또는 상수이다.

그림 15-12 함수 fprintf()와 fscanf() 함수원형

표 15-1 표준 파일의 종류

표준 파일	키워드	장치(device)
표준입력	stdin	키보드
표준출력	stdout	모니터 화면
표준에러	stderr	모니터 화면

표준입력 자료를 파일에 쓰기

- **파일에 내용 쓰기와 반대로 파일로부터 내용을 읽는 프로그램을 작성**
  - 쓰기 모드로 파일을 열어
    - 표준 입력으로 받은 학생이름과 중간점수, 기말점수를 파일에 기록하고 파일을 닫기
  - 다시 읽기 모드로 그 파일을 열어 기록된 내용을 읽어와
    - 표준출력으로 출력하는 프로그램
  - 함수 scanf_s()에서 표준입력을 빈칸으로 구분하므로 빈칸이 없는 이름을 입력
  - 학생이름, 중간점수, 기말점수를 함수 fprintf()를 이용하여 파일 f에 출력
    - 변수 cnt는 이름 앞에 번호를 붙이려는 목적으로 이용

```
scanf_s("%s%d%d", name, 30, &point1, &point2);
                  ↑ 문자열이 저장되는 name과 그 크기를 지정해야 한다.

fprintf(f, "%d %s %d %d\n", ++cnt, name, point1, point2);
```

그림 15-13 함수 scanf_s()와 fprintf()의 이용

  - 파일 f에서 함수 fscanf_s()를 이용하여 구조체의 정보를 입력하려면
    - 파일의 내부 자료의 저장 형태를 알아야 함
  - 학생이름, 중간, 기말이므로 다음과 같이 문장으로
    - fscanf_s()를 사용

```
                  문자열이 저장되는 name과 그 크기를 지정해야 한다.
                  ↓
//파일"grade.txt"에서 읽기
fscanf_s(f, "%d %s %d %d\n", &cnt, name, 30, &point1, &point2);
```

그림 15-14 함수 fscanf_s()의 이용

파일에 쓰기와 읽기

예제 fprintf.c

- 함수 fscanf()와 fprintf()

실습예제 15-2 fprintf.c

이름, 성적을 입력하여 간단한 파일을 생성 후 다시 읽어보는 프로그램

```c
// file: fprintf.c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

int main()
{
   char fname[] = "grade.txt";
   FILE *f;
   char name[30];
   int point1, point2, cnt = 0;

   if (fopen_s(&f, fname, "w") != 0)
   //if ( (f = fopen(fname, "w")) == NULL )
   {
      printf("파일이 열리지 않습니다.\n");
      exit(1);
   };
   printf("이름과 성적(중간, 기말)을 입력하세요.\n");
   scanf("%s %d %d", name, &point1, &point2);
   //scanf_s("%s%d%d", name, 30, &point1, &point2);
   //파일 "grade.txt"에 쓰기
   fprintf(f, "%d %s %d %d\n", ++cnt, name, point1, point2);
   fclose(f);

   if (fopen_s(&f, fname, "r") != 0)
   //if ( (f = fopen(fname, "r")) == NULL )
   {
      printf("파일이 열리지 않습니다.\n");
      exit(1);
   };
   //파일"grade.txt"에서 읽기
   fscanf(f, "%d %s %d %d\n", &cnt, name, &point1, &point2);
   //fscanf_s(f, "%d %s %d %d\n", &cnt, name, 30, &point1, &point2);
   //표준출력에 쓰기
   fprintf(stdout, "\n%6s%16s%10s%8s\n", "번호", "이름", "중간", "기말");
   fprintf(stdout, "%5d%18s%8d%8d\n", cnt, name, point1, point2);
   fclose(f);

   return 0;
}
```

함수 fprintf(stdout, …)는 함수 printf(…)를 이용할 수도 있다.

23 파일 grade.txt에 쓰기 위해 함수 fprintf() 사용, 첫 인자는 파일 포인터 f이며, 나머지는 printf()와 동일하므로 파일에 입력번호(++cnt), 이름(name), 중간(point1), 기말(point2)을 씀

24 파일 읽기가 종료되었으면 fclose()로 파일 닫기

26 다시 파일 grade.txt을 읽기 위해 파일 열기

33 이미 파일 grade.txt에 쓴 입력번호, 이름, 중간, 기말을 각각 변수 cnt, name, point1, point2에 읽어 오기 위해 함수 fscanf() 사용

36 콘솔에 제목 줄 출력

37 26행에서 읽어 온 자료를 fprintf() 사용하여 표준출력

38 파일 읽기가 종료되었으면 fclose()로 스트림을 닫음

변수 fname에 생성할 파일이름인 "grade.txt"를 저장
파일포인터 f를 하나 선언
이름을 저장할 변수 name 선언
중간고사(point1), 기말고사(point2), 입력 학생수(cnt)를 위한 변수 선언
함수 fopen()의 호출 시, 첫 번째 인자는 파일 포인터의 주소, 두 번째 인자는 파일이름, 세 번째 인자는 모드로 "w"는 쓰기모드이며, 반환값은 오류 수로 0이 아니면 파일 열기에 문제가 발생한 것
파일 열기에 실패하면 메시지 출력하고 exit(1)으로 종료
표준입력으로 이름, 중간, 기말 성적을 받아 10, 11행에서 선언된 변수 name, point1, point2에 저장

실행결과

```
이름과 성적(중간, 기말)을 입력하세요.
김소현 90 98

   번호            이름      중간    기말
    1            김소현        90      98
```

함수 fgets()와 fputs()

- **함수 fgets(): 파일로부터 한 행의 문자열을 입력 받는 함수**
  - 파일로부터 문자열을 개행문자(₩n)까지 읽어 마지막 개행문자를 '₩0'문자로 바꾸어 입력 버퍼 문자열에 저장
    - 첫 번째 인자는 문자열이 저장될 문자 포인터
    - 두 번째 인자는 입력할 문자의 최대 수, 세 번째 인자는 입력 문자열이 저장될 파일
- **함수 fputs(): 파일로 한 행의 문자열을 출력하는 함수**
  - 문자열을 한 행에 출력
    - 첫 번째 인자는 출력될 문자열이 저장된 문자 포인터
    - 두 번째 인자는 문자열이 출력되는 파일
- **fgets()와 fputs()는 헤더파일 stdio.h 필요**

함수 fgets()와 fputs() 함수원형

```
char * fgets(char * _Buf, int _MaxCount, FILE * _File);
int  fputs(char * _Buf, FILE * _File);
```

- 함수 fgets()는 _File로부터 한 행의 문자열을 _MaxCount 수의 _Buf 문자열에 입력 수행
- 함수 fputs()는 _Buf 문자열을 _File에 출력 수행

```
char names[80];
FILE *f;

fgets(names, 80, f);
fputs(names, f);
```

그림 15-15 함수 fgets()와 fputs() 함수원형

함수 feof()와 ferror()

- **함수 feof(): 파일 스트림의 EOF(End Of File) 표시를 검사하는 함수**
  - 읽기 작업이 파일의 이전 부분을 읽으면 0을 반환하고
    - 그렇지 않으면 0이 아닌 값을 반환
    - 파일 스트림의 EOF은 이전 읽기 작업에서 EOF 표시에 도달하면 0이 아닌 값으로 지정
  - 단순히 파일 지시자가 파일의 끝에 있더라도 feof()의 결과는 0
- **함수 ferror(): 파일 처리에서 오류가 발생했는지 검사하는 함수**
  - 이전 파일 처리에서 오류가 발생하면 0이 아닌 값을 반환
    - 오류가 발생하지 않으면 0을 반환
  - 헤더파일 stdio.h 필요

함수 feof()와 ferror() 함수원형

```
int   feof(FILE * _File);
int   ferror(FILE * _File);
```

- 함수 feof()은 _File의 EOF를 검사
- 함수 ferror()는 _File 에서 오류발생 유무를 검사

```
while ( !feof(stdin) )
{
   ...
   fgets(names, 80, stdin);    //표준입력
}
```

그림 15-16 함수 feof()와 ferror() 함수원형

여러 줄의 입력

예제 mlineio.c

- 함수 fgets()와 fputs()를 이용하여 표준입력으로 여러 줄을 입력 받아 여러 줄을 파일 grade.txt에 출력
- 여러 줄의 표준입력을 처리하기 위하여 while (!feof(stdin)) {...} 구문을 이용
- 함수 fputs()를 이용하기 전에 함수 fprintf()를 이용하여 줄 번호를 출력
- 파일 grade.txt 저장 시 맨 앞에 1부터 순차적으로 번호를 삽입
- 표준입력에서 입력을 종료하려면 키 ctrl+Z를 새로운 행의 처음에 누름

실습예제 15-3 mlineio.c

여러 줄에 걸쳐 이름, 성적을 입력하여 파일에 그 내용을 모두 저장

```
01  // file: mlineio.c
02  #define _CRT_SECURE_NO_WARNINGS
03  #include <stdio.h>
04  #include <stdlib.h>
05
06  int main()
07  {
08     char fname[] = "grade.txt";
09     FILE *f;
10     char names[80];
11     int cnt = 0;
12
13     //if ( (f = fopen(fname, "w")) == NULL )
14     if (fopen_s(&f, fname, "w") != 0)
15     {
16        printf("파일이 열리지 않습니다.\n");
17        exit(1);
18     };
19     printf("이름과 성적(중간, 기말)을 입력하세요.\n");
20     fgets(names, 80, stdin);
21
22     //콘솔에 이름 중간 기말 입력하고 Enter 키
23     //여러 줄에 입력하다가
24     //종료하고 싶을 때 새줄 첫 행에서 ctrl + Z 누름
25     while ( !feof(stdin) )
26     {
27        //파일 "grade.txt"에 쓰기
28        fprintf(f, "%d ", ++cnt);   //맨 앞에 번호를 삽입
29        fputs(names, f);   //이후에 입력 받은 이름과 성적 2개 저장
30        fgets(names, 80, stdin);   //다시 표준입력
31     }
32     fclose(f);
33
34     return 0;
35  }
```

설명

```
08    fname에 생성할 파일이름을 저장
09    파일포인터를 하나 선언
10    파일에 쓸 자료값으로 한 행을 저장할 char 배열
11    파일에 쓸 자료값으로 행 마다 행 번호를 저장할 변수
13    함수 fopen_s()의 호출 시, 첫 인자는 파일 포인터의 주소, 두 번째 인자는 파일이름, 세 번째
      인자는 모드로 "w"는 쓰기모드이며, 반환값인 정수가 0이 아니면 파일 열기에 실패
16~17 파일 열기에 실패하면 메시지 출력하고 exit(1)으로 종료
20    표준입력으로 받은 한 행을 변수 names에 저장, 한 행이 79 열보다 크면 배열 names[]의 크기를
      더 크게 조정
25    표준입력이 있으면 계속, ctrl+Z 로 feof()이면 종료
28    파일 grade.txt에 행마다 행 번호를 출력
29    파일 grade.txt에 배열 names[]을 내용을 출력, 첫 인자는 출력한 내용인 names이며,
      두 번째 인자가 파일 포인터 f,
30    다시 표준입력에서 한 행을 변수 names에 저장
32    파일 처리가 종료되었으면 fclose()로 스트림을 닫음
```

실행결과

```
이름과 성적(중간, 기말)을 입력하세요.
염보라 87 95
안봉선 78 96
박지혜 85 87
^Z
```

표준 입력으로 여러 줄에 걸쳐 적당한 형태로 입력하고 마지막 행에는 반드시 키 ctrl + Z를 입력한다.

함수 fgetc()와 fputc()

- **함수 fgetc()와 getc()**
  - 파일로부터 문자 하나를 입력받는 함수
- **함수 fputc()와 putc()**
  - 문자 하나를 파일로 출력하는 함수
  - 함수들은 문자 하나의 입출력의 대상인 파일 포인터를 인자로 이용
- **헤더파일 stdio.h 필요**

함수 fgetc()와 fputc() 함수원형

```
int   fgetc(FILE * _File);
int   fputc(int _Ch, FILE * _File);

int   getc(FILE * _File);
int   putc(int _Ch, FILE * _File);
```

- 함수 fgetc()아 getc()는 _File에서 문자 하나를 입력받는 함수
- 함수 fputc()와 putc()문자 _Ch를 파일 _File 에 출력하는 함수

그림 15-18 함수 fgetc()와 fputc() 함수원형

- **getchar()와 putchar()**
  - getc()와 putc()를 이용한 매크로로 정의
  - getchar()와 putchar()는 함수로도 구현

```
#define  getchar()          getc(stdin)
#define  putchar(_c)        putc((_c),stdout)

int   getchar(void);
int   putchar(int _Ch);
```

문자 입출력

예제 fgetc.c

- 여러 문자를 표준입력으로 받아 파일 char.txt에 저장한 후, 다시 파일에서 문자를 읽어 표준출력하는 프로그램
- 프로그램 실행
  - 프로젝트 폴더에 파일 char.txt가 생성되고
  - x를 입력하기 전까지의 문자가 입력

실습예제 15-4 fgetc.c

여러 문자를 표준입력으로 받아 파일에 저장한 후, 다시 파일에서 문자를 읽어 표준출력

```
01  // file: fgetc.c
02  #define _CRT_SECURE_NO_WARNINGS
03  #include <stdio.h>
04  #include <stdlib.h>
05  #include <conio.h>
06
07  int main()
08  {
09      char fname[] = "char.txt"; //입력한 내용이 저장될 파일이름
10      FILE *f; //파일 포인터
11
12      //쓰기모드로 파일 열기
13      if (fopen_s(&f, fname, "w") != 0)
14      //if ( (f = fopen(fname, "w")) == NULL )
15      {
16          printf("파일이 열리지 않습니다.\n");
17          exit(1);
18      };
19      puts("문자를 입력하다가 종료하려면 x를 입력 >>");
20
21      /*표준입력으로 받은 문자를 파일에 출력하는 부분*/
22      int ch; //입력된 문자 저장
23      while ((ch = _getche()) != 'x')
24          //파일 "char.txt"에 쓰기
25          fputc(ch, f);   //파일에 문자 출력
26      fclose(f); puts("");
27
28      //읽기모드로 파일 열기
29      if (fopen_s(&f, fname, "r") != 0)
30      {
31          printf("파일이 열리지 않습니다.\n");
32          exit(1);
33      };
34
35      /*파일에서 다시 문자를 입력받아 콘솔에 표준출력하는 부분*/
36      while ((ch = fgetc(f)) != EOF)
37          //파일 "char.txt"에서 다시문자 읽기
38          _putch(ch);   //파일로부터 입력 받은 문자를 표준출력
39      fclose(f); puts("");
40
41      return 0;
42  }
```

설명

```
09  변수 fname에 생성할 파일이름인 "char.txt"을 저장
10  파일포인터 f를 하나 선언
13  함수 fopen()의 호출 시, 첫 번째 인자는 파일 포인터의 주소, 두 번째 인자는 파일이름, 세 번째
    인자는 모드로 "w"는 쓰기모드이며, 반환값은 오류 수로 0이 아니면 파일 열기에 문제가 발생한 것
22  표준입력으로 받은 문자 하나를 저장할 변수 ch 선언
23  함수 _getche()로 문자를 하나 표준입력 받아 ch에 저장, ch의 문자가 x가 아니면 while 반복
    계속
25  함수 fputc()로 파일 f에 바로 전에 입력 받은 문자 ch를 저장
26  파일 쓰기가 종료되었으면 fclose()로 파일 닫기
29  다시 파일 char.txt을 읽기 위해 파일 열기
36  함수 fgetc()로 파일에서 문자를 하나 읽어 변수 ch에 저장, ch가 EOF이 아니
    반복 실행
38  함수 _putch()로 문자 ch를 표준 출력
39  파일 읽기가 종료되었으면 fclose()로 파일 닫기
```

실행결과

```
문자를 입력하다가 종료하려면 x를 입력 >>
fopen_s(&f, fname, 'w') x
fopen_s(&f, fname, 'w')
```

파일 내용을 표준출력으로 그대로 출력

- **도스 명령어 type와 같이**
  - 파일의 내용을 그대로 콘솔에 출력하는 프로그램 list
    - "list filename"을 입력
      - 파일 filename의 내용을 표준출력하는 프로그램
      - 명령행 인자에서 두 번째 인자가 파일이름에 해당
    - 파일 내용의 출력은 한 줄마다 맨 앞에 줄 번호를 출력
- **실행**
  - 실행파일인 list.exe는 폴더 [Ch15/Debug]에 있으므로
    - 도스 프롬프트에서 폴더 [Ch15/Debug]로 이동하여 실행
  - 물론 실행 이전에 반드시 listpart.c 파일을 먼저 생성 후 실행

```
D:\Creative C Sources\Ch15\Debug>list ../Prj05/listpart.c
 1: // file: listpart.c
 2:
 3: #include <stdio.h>
 4: #include <stdlib.h>
 5:
 6: int main(int argc, char *argv[])
 7: {
 8:    FILE *f;
 9:    int ch, cnt = 0;
10:    ...
11:    return 0;
12: }
```

두 번째 인자인 "../list/listpart.c"를 파일 이름으로 이 파일의 내용을 출력한다.

줄 번호와 함께 파일 내용을 그대로 출력한다.

그림 15-19 예제 list를 도스 프롬프트에서 실행한 결과

파일 내용 표시

예제 list.c

- 텍스트 파일의 내용을 모도 출력
  - Visual C++에서 위 프로그램을 실행하려면 메뉴 [프로젝트/속성]을 선택한 대화상자에서 [명령인수]에 화면에 표시할 파일을 기술
  - 위 소스인 파일 list.c를 입력하면 소스 전체를 행 번호와 함께 콘솔에 표시

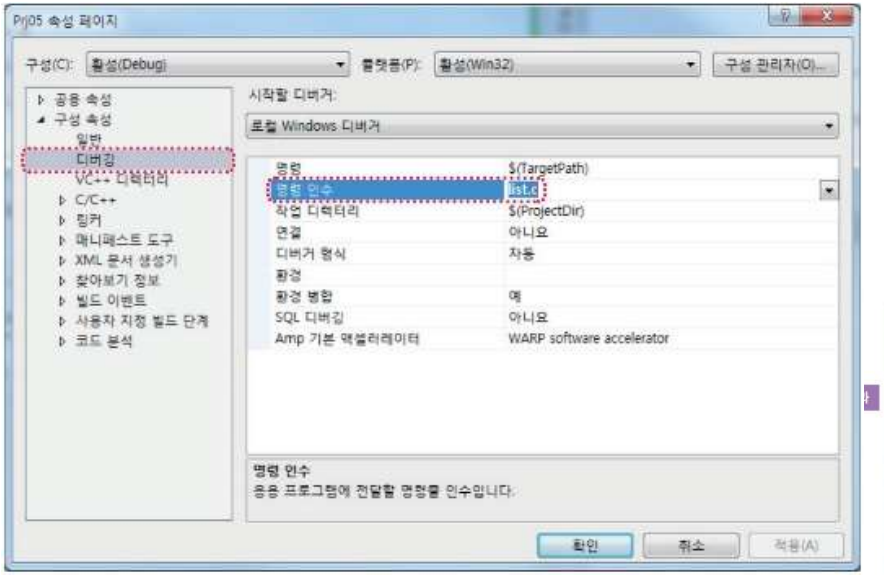


그림 15-20 Visual C++에서 명령행 인자를 기술하는 방법

실습예제 15-5 list.c

명령행 인자를 사용해서 파일의 내용을 표준출력으로 그대로 출력하는 프로그램

```
01  // file: list.c
02  #include <stdio.h>
03  #include <stdlib.h>
04
05  int main(int argc, char *argv[])
06  {
07     FILE *f;
08
09     if (argc != 2)
10     {
11        printf("사용법: list filename\n");
12        exit(1);
13     }
14
15     //읽기 모드로 파일 열기
16     if (fopen_s(&f, argv[1], "r") != 0)
17     //if ( (f = fopen(argv[1], "r")) == NULL )
18     {
19        printf("파일이 열리지 않습니다.\n");
20        exit(1);
21     }
22
23     //문자를 저장할 ch, 행번호를 저장할 cnt
24     int ch, cnt = 0;
25     printf("%4d: ", ++cnt);
26     while ((ch = fgetc(f)) != EOF)
27     {
28        putchar(ch); //putc(ch, stdout);
29        //행 처음에 행 번호 출력
30        if (ch == '\n') printf("%4d: ", ++cnt);
31     }
32     printf("\n");
33     fclose(f);
34
35     return 0;
36  }
```

설명

```
07  파일포인터를 하나 선언
09  명령행 인자에서 출력할 파일이름이 없으면 사용법을 출력
11  사용법을 출력
12  프로그램 종료
16  읽기 모드로 파일 열기
25  처음에 행 번호 1 출력
26  파일 f에서 문자 하나를 입력해 ch에 저장, EOF이면 while 반복 종료
```

```
28  표준출력에 문자 ch 출력
30  새로운 줄에 이동했으면 다시 처
33  파일 처리가 종료되었으면 fclos
```

```
 1: // file: list.c
 2: #include <stdio.h>
 3: #include <stdlib.h>
 4:
 5: int main(int argc, char *argv[])
 6: {
 7:    FILE *f;
 8:
 9:    if (argc != 2)
10:    {

중간생략

33:    fclose(f);
34:
35:    return 0;
36: }
37:
```

LAB 파일의 대소문자를 서로 바꿔 다른 파일로 생성

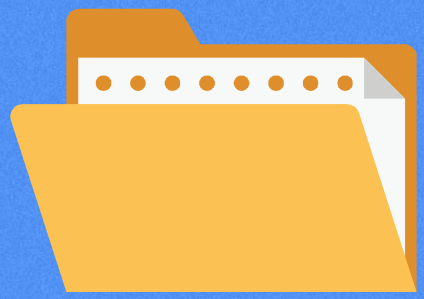
- **소스 파일 convertchar.c의 알파벳에서**
  - 대문자는 소문자로, 소문자는 대문자로 변환하여 새로운 파일 my_convertchar.c에 출력하는 프로그램
    - 두 파일을 각각 열기모드와 쓰기모드로 열어
    - 함수 fgetc() 또는 getc()로 문자 하나를 읽어 알파벳이면 대소문자로 바꿔 다시 함수 fputc() 또는 putc()로 변환된 문자를 출력
    - 문자 하나를 읽어 그 문자가 EOF이 아니면 계속 변환하여 출력
  - 문자 처리에 오류가 발생해 소스에서 한글은 없도록
- **결과**
  - File my_convertchar.c is created!!!

Lab 15-2 convertchar.c

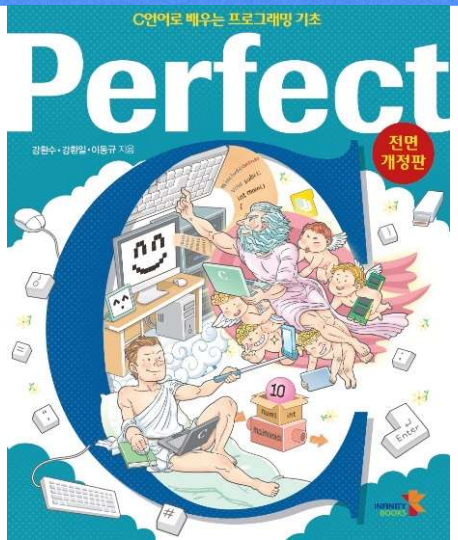
```
01  // file: convertchar.c
02  #define _CRT_SECURE_NO_WARNINGS
03  #include <stdio.h>
04  #include <stdlib.h>
05  #include <ctype.h>
06
07  int main(void)
08  {
09     FILE *f1, *f2;
10     if ((f1 = fopen("convertchar.c", "r")) == NULL) {
11        printf("cannot open this file\n");
12        exit(1);
13     }
14     if ((f2 = fopen(_____________________________________)) == NULL) {
15        printf("cannot open this file\n");
16        fclose(f1);
17        exit(1);
18     }
19
20     char a;
21     while ((a = _________________) != _______________)
22     {
23        if (isalpha(a))
24           if (islower(a))
25              a = toupper(a);
26           else if (isupper(a))
27              a = tolower(a);
28        putc(_______);
29     }
30
31     fclose(f1);
32     fclose(f2);
33     printf("File my_convertchar.c is created!!!\n");
34
35     return 0;
36  }
```

정답

```
14  if ((f2 = fopen("my_convertchar.c", "w")) == NULL) {
21  while ((a = getc(f1)) != EOF)
28  putc(a, f2);
```



03. 이진 파일 입출력



함수 fprinf()와 fsanf_s()

- **자료의 입출력을 텍스트 모드로 처리**
  - 출력된 텍스트 파일은 텍스트 편집기로 그 내용을 볼 수 있으며
  - 텍스트 파일의 내용은 모두 지정된 아스키 코드와 같은 문자 코드값
    - 그 내용을 확인할 수 있을 뿐만 아니라 인쇄 가능
- **함수 fprintf()를 이용**
  - int 형 변수 cnt의 값을 파일 f에 출력하는 과정
  - 실제로 파일에 저장되는 자료는 정수값 10에 해당하는 각 문자의 아스키 값
    - 각각의 문자 '1'과 '0'의 아스키 코드값이 저장

```
int cnt = 10;

fprintf(f, "%d", cnt);
```

그림 15-21 함수 fprintf()에 의한 텍스트 파일 출력의 예

함수 fwrite()와 fread()

- **함수 fwrite()와 fread()**
  - 이진파일(binary file)은 C 언어의 자료형을 모두 유지하면서 바이트 단위로 저장되는 파일
  - 이진 모드로 블록 단위 입출력을 처리
  - 헤더파일 stdio.h 필요

함수 fwrite()와 fread() 함수원형

```
size_t  fwrite(const void *ptr, size_t size, size_t n, FILE *f);
size_t  fread(void *dstbuf, size_t size, size_t n, FILE *f);
```

- 함수 fwrite()는 ptr이 가리키는 메모리에서 size만큼 n개를 파일 f에 쓰는(저장) 함수
- fread()는 반대로 파일 f에서 elmtsize의 n개만큼 메모리 dstbuf에 읽어오는 함수, 반환값은 성공적으로 입출력을 수행한 항목의 수

```
int cnt = 10;
fwrite(&cnt, sizeof(int), 1, f);
fread(&cnt, sizeof(int), 1, f);
```

그림 15-22 함수 fwrite()와 fread() 함수원형

- **함수 fwrite**
  - 첫 번째 인자 ptr은 출력될 자료의 주소값
  - 두 번째 인자 size는 출력될 자료 항목의 바이트 크기
  - 세 번째 인자는 출력될 항목의 개수이며, 마지막 인자는 출력될 파일 포인터
  - 파일 f에 ptr에서 시작해서 size*n 바이트만큼의 자료를 출력
    - 반환값은 출력된 항목의 개수
- **함수 fread()**
  - 이진 파일에 저장되어 있는 자료를 입력
  - 함수 fwrite()와 인자는 동일

함수 fwrite()를 이용 과정

- **함수 fwrite()**
  - 바이트 단위로 원하는 블록을 파일에 출력하기 위한 함수
  - 출력된 자료는 함수 fread()로 입력해야 그 자료유형을 유지
  - 세 번째 항목인 출력 항목 수를 4로 지정한 경우의 출력
- **이진 파일을 위한 파일 열기 모드**
  - 문자 'b'를 추가

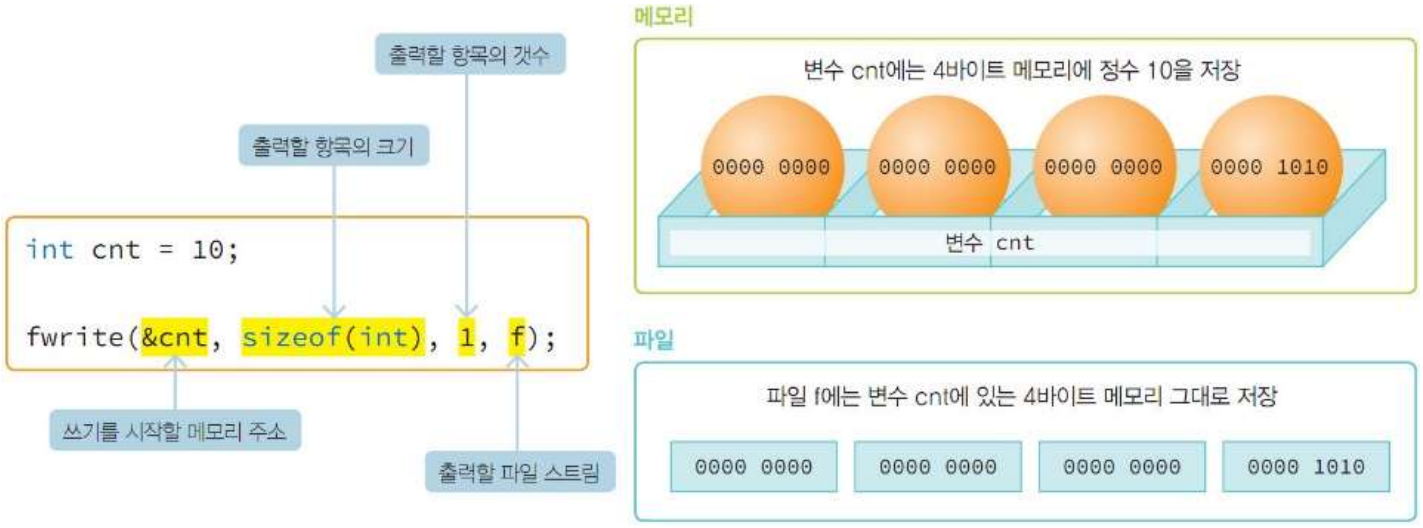


그림 15-23 함수 fwrite()에 의한 이진 파일 출력의 예

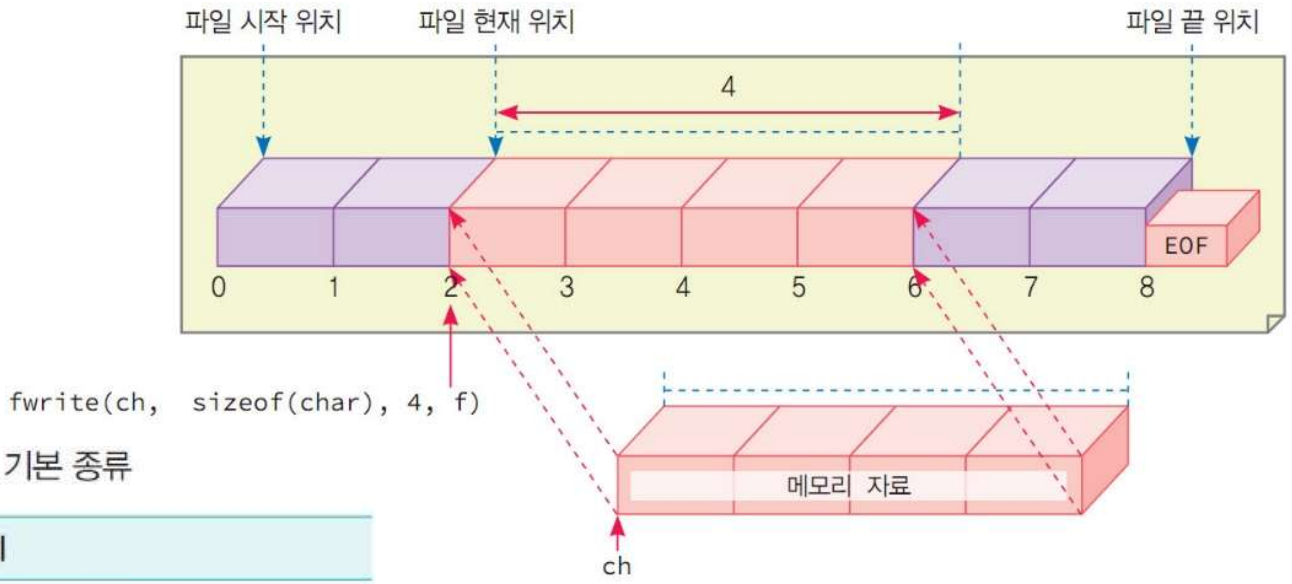


그림 15-24 블록 쓰기 함수 fwrite()

표 15-2 이진파일 열기 함수 fopen()의 모드 기본 종류

모드	의미
rb	이진파일의 읽기(read) 모드로 파일을 연다.
wb	이진파일의 쓰기(write) 모드로 파일을 연다.
ab	이진파일의 추가(append) 모드로 파일을 연다.

학생 성적 구조체 파일 쓰기

- **구조체 personscore: 학생의 성적정보를 구조체로 표현**
  - 번호, 이름, 중간, 기말, 퀴즈 점수를 멤버로 구성

```
struct personscore
{
    int number;          //번호
    char name[40];       //이름
    int mid;             //중간성적
    int final;           //기말성적
    int quiz;            //퀴즈성적
};
typedef struct personscore pscore;
```

- **표준입력으로 여러 명의 자료를 입력 받은 구조체 자료형을 파일 "score.bin"에 저장하는 프로그램**
  - 표준입력은 사람마다 한 행씩 입력 받도록
    - 입력된 학생 수로 학생 번호를 입력
  - 행마다 fgets()를 이용하여 하나의 문자열로 받고
    - 입력된 문자열에서 각 구조체의 멤버 자료를 추출
    - 문자열에서 자료를 추출하기 위하여 함수 sscanf()를 이용

구조체를 저장하는 이진 파일

예제 ptrtypecast.c

- 이진파일 score.bin이 프로젝트 폴더에 생성
  - 그 내용을 직접 볼 수 없음
  - 물론 일부 문자열 자료는 그대로 아스키 코드값으로 저장되고 출력되므로 그 내용을 확인

실습예제 15-6 fwrite.c

여러 줄에 걸쳐 학생 구조체의 정보를 입력하여 파일을 생성하는 프로그램

```
01  // file: fwrite.c
02  #define _CRT_SECURE_NO_WARNINGS
03  #include <stdio.h>
04  #include <string.h>
05  #include <stdlib.h>
06
07  struct personscore //구조체 struct personsc
```

실행결과

```
이름과 성적(중간, 기말, 퀴즈)을 입력하세요.
안혜지 89 76 98
조손정 76 87 59
김혜원 98 89 99
^Z
```

```
08  {
09      int number;       //번호
10      char name[40];    //이름
11      int mid;        //중간성적
12      int final;        //기말성적
13      int quiz;       //퀴즈성적
14  };
15  typedef struct personscore pscore; //구조체 자료형 pscore 정의
16
17  int main()
18  {
19      char fname[] = "score.bin";
20      FILE *f;
21
22      //쓰기 모드로 파일 열기
23      if (fopen_s(&f, fname, "wb") != 0)
24      //if ( (f = fopen(fname, "wb")) == NULL )
25      {
26          printf("파일이 열리지 않습니다.\n");
27          exit(1);
28      };
29
30      //표준입력으로 행을 저장하기 위한 변수
31      char line[80];
32      int cnt = 0; //입력 학생 번호(자동 생성) 변수
33      pscore score; //구조체 변수 선언
34      printf("이름과 성적(중간, 기말, 퀴즈)을 입력하세요.\n");
35
36      fgets(line, 80, stdin);
37      while (!feof(stdin))
38      {
39          //표준입력의 한줄을 구조체의 멤버 별로 자료를 입력
40          //sscanf(line, "%s %d %d %d", score.name, &score.mid, &score.final,
            &score.quiz);
41          sscanf_s(line, "%s %d %d %d", score.name, 40,
42              &score.mid, &score.final, &score.quiz);
43          score.number = ++cnt;
44          fwrite(&score, sizeof(pscore), 1, f);
45          fgets(line, 80, stdin);
46      }
47      fclose(f);
48
49      return 0;
50  }
```

파일에서 학생 성적 구조체 읽기(1)

- **예제에서 만든 이진 파일 score.bin의 내용을 읽어 표준 출력하는 프로그램을 작성**
  - 함수 feof()을 이용하여 파일의 마지막까지 구조체 자료를 읽어
    - 적당한 출력 형태가 되도록 함수fprintf()와 prinf()를 이용하여 출력
  - 예제를 실행하려면 위 예제에서 생성된 파일 score.bin을 반드시 이 프로젝트 하부 폴더에 복사

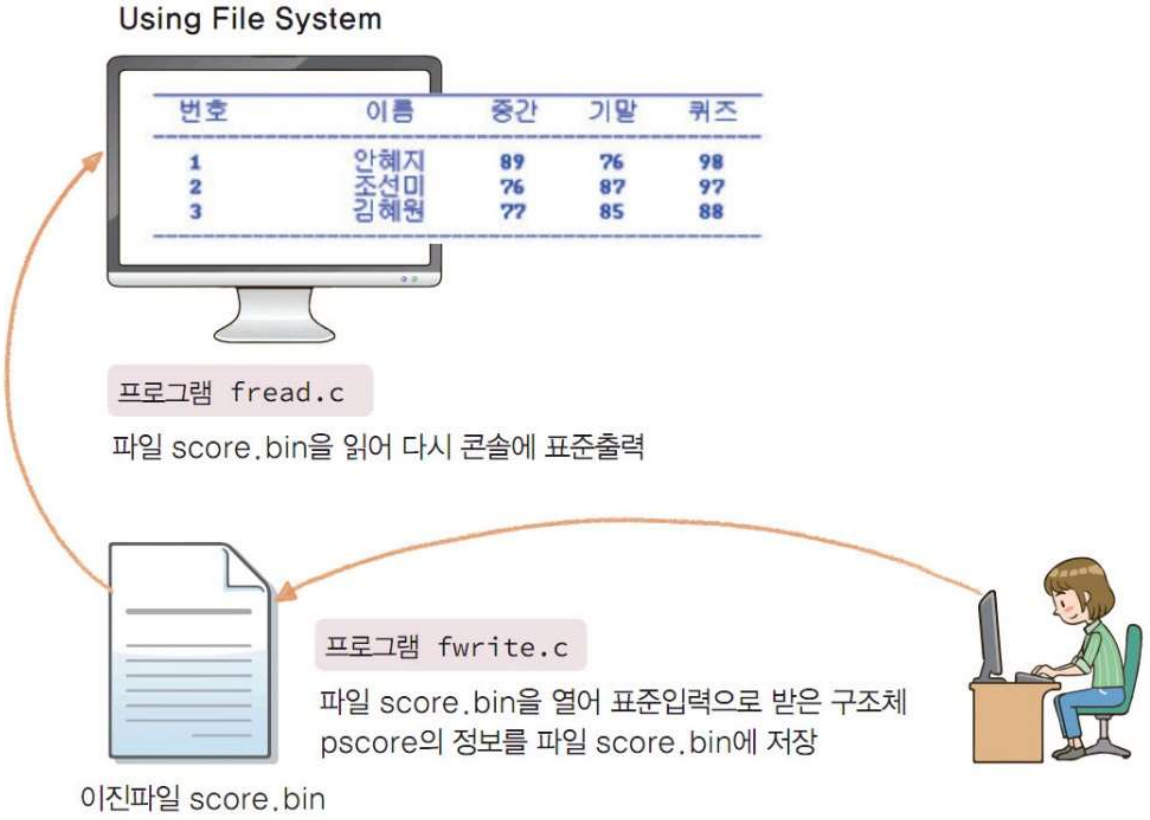


그림 15-25 구조체 pscore의 변수를 파일 score.bin에 저장한 후 다시 score.bin 파일을 읽어 표준출력

파일에서 학생 성적 구조체 읽기(2)

예제 fread.c

실습예제 15-7 fread.c

학생 성적 구조체 정보가 저장된 파일을 읽어 표준출력에 출력

```c
// file: fread.c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

struct personscore
{
   int number;
   char name[40];
   int mid;
   int final;
   int quiz;
};
typedef struct personscore pscore;
void printhead();

int main()
{
   char fname[] = "score.bin";
   FILE *f;

   if ((f = fopen(fname, "rb")) == NULL)
   //if (fopen_s(&f, fname, "rb") != 0)
   {
      printf("파일이 열리지 않습니다.\n");
      exit(1);
   };
   printhead();

   //이진모드로 파일 f에서 구조체 pscore 자료 읽기
   pscore score;
   fread(&score, sizeof(pscore), 1, f);
   while ( !feof(f) )
   {
      //표준출력에 쓰기
      fprintf(stdout, "%6d%18s%8d%8d%8d\n",
         score.number, score.name, score.mid, score.final, score.quiz);
      fread(&score, sizeof(pscore), 1, f);
   }
   printf("%s\n", "  ------------------------------------------------");
   fclose(f);

   return 0;
}

void printhead()
{
   printf("%s\n", "  ------------------------------------------------");
   printf("%8s%15s%10s%8s%8s\n", "번호", "이름", "중간", "기말", "퀴즈");
   printf("%s\n", "  ------------------------------------------------");
}
```

설명

- 07~14 구조체 struct personscore 정의
- 15 구조체 struct personscore를 자료형 pscore로 정의
- 20 fname에 읽을 파일이름인 score.bin 저장
- 21 파일포인터를 하나 선언
- 23 함수 fopen()의 호출 시, 첫 번째 인자는 파일이름, 두 번째 인자는 모드로 "rb"는 이진으로 읽기 모드이며, 반환값을 파일 포인터에 저장
- 29 제목출력 함수 printhead() 호출
- 32 이진모드로 파일 f에서 구조체 pscore 자료를 읽어 저장할 변수 선언
- 33 이진모드로 파일 f에서 구조체 pscore 자료를 하나 읽어 변수 score에 저장
- 34 파일 f가 EOF가 아니면 반복 계속
- 37 표준출력으로 구조체 pscore 자료인 score를 출력
- 39 다시 이진모드로 파일 f에서 구조체 pscore 자료를 하나 읽어 변수 score에 저장
- 42 파일 처리가 종료되었으면 fclose()로 스트림을 닫음

실행결과

```
------------------------------------------------
  번호            이름      중간      기말      퀴즈
------------------------------------------------
     1            안혜지      89        76        98
     2            조손정      76        87        59
     3            김혜원      98        89        99
------------------------------------------------
```

LAB
학생정보를 표현하는 구조체 배열을 파일에 쓰고 다시 파일에 저장된 학생정보를 다시 읽어 오기

- **학생 정보를 표현하는 구조체 student**
  - 학과와 이름, 학번으로 구성
- **학생 3명의 정보를 초기화로 배열 mylab에 저장한 후**
  - 파일 student.bin을 쓰기 모드로 열어 함수 fwrite()로 배열을 모두 출력
  - 다시 파일 student.bin을 열기 모드로 열어
    - 저장할 구조체 배열 lab을 선언하여 함수 fread()로 배열을 모두 입력
  - 배열 lab에 저장된 학생 정보를 다시 콘솔에 출력
  - 함수 fwrite(자료 주소값, 하나의 바이트 크기, 총 개수, 출력파일 포인터)
    - 첫 번째 인자 ptr은 출력될 자료의 주소값이며, 두 번째 인자 size는 출력될 자료 항목의 바이트 크기이고, 세 번째 인자는 출력될 항목의 개수이며, 마지막 인자는 출력될 파일 포인터
  - 함수 fread(입력자료 주소값, 하나의 바이트 크기, 총 개수, 입력파일 포인터)
    - 첫 번째 인자는 저장될 버퍼 자료의 주소값이며, 두 번째 인자 size는 입력될 자료 항목의 바이트 크기이고, 세 번째 인자는 입력될 항목의 개수이며, 마지막 인자는 입력 파일 포인터
- **결과**
  - 컴퓨터정보공학과 김하늘 201698657
  - 컴퓨터정보공학과 백규정 201648762
  - 컴퓨터소프트웨어공학과 김효주 201665287

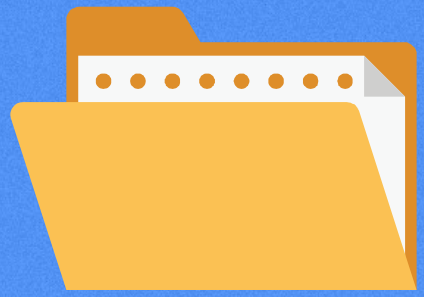
LAB 계속

Lab 15-3 studentinfo.c

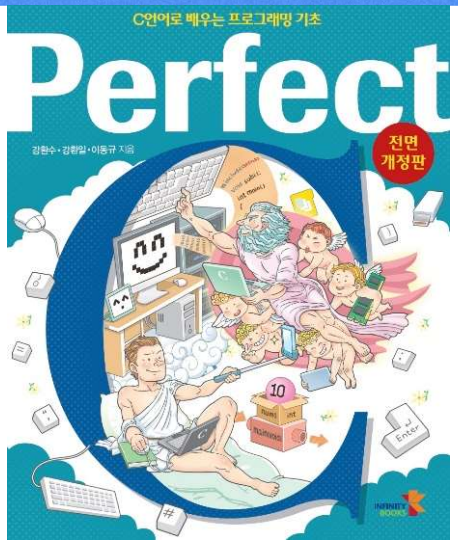
```c
//file: studentinfo.c
#include <stdio.h>

//구조체 자료형 student 정의
typedef struct student
{
   char dept[40];   //학과
   char name[20];   //이름
   int snum;   //학번
} student;

int main()
{
   student mylab[] = {
      { "컴퓨터정보공학과", "김하늘", 201698657},
      { "컴퓨터정보공학과", "백규정", 201648762 },
      { "컴퓨터소프트웨어공학과", "김효주", 201665287 } };

   FILE *f;
   char fname[] = "student.bin";
   fopen_s(&f, fname, "wb");
   int size = _____________________________________________;
   fwrite(mylab, _____________________________________________);
   fclose(f);

   //다시 읽기 위해 오픈
   fopen_s(&f, fname, "rb");
   //파일에서 구조체 배열 모두를 한번에 읽어 다시 저장된 배열을 출력
   student lab[10]; //다시 파일의 내용을 저장할 배열 선언
   //파일 f에서 sizeof(student) 크기로 size 수만큼 읽어 lab에 저장
   fread(lab, _____________________________________________);
   for (int i = 0; i < size; i++)
      fprintf(stdout, "%24s%10s%12d\n", _____________________________);
   fclose(f);

   return 0;
}
```

정답

```c
int size = sizeof(mylab) / sizeof(student);
fwrite(mylab, sizeof(student), size, f);
fread(lab, sizeof(student), size, f);
   fprintf(stdout, "%24s%10s%12d\n", lab[i].dept, lab[i].name, lab[i].snum);
```



04. 파일 접근 처리



파일 위치

- **파일 위치는 파일 내부를 바이트 단위로 파일 내부 위치를 나타내는 값**
  - '파일 지시자(file indicator)' 또는 '파일 표시자'라고도 부름
  - 파일의 시작점에서 파일 위치는 0이며 1바이트마다 1씩 증가
  - 파일의 마지막에는 파일의 마지막임을 알리는 EOF(End Of File) 표시
- **파일을 열면**
  - 파일 위치(file position)는 항상 파일의 시작 부분을 가리킴

그림 15-26 파일 위치의 의미

- **만일 파일 위치가 100L**
  - 파일의 처음에서부터 100바이트 떨어진 위치에 현재 파일 위치
  - 파일 위치 값은 일반적으로 자료형 long으로 취급
    - 상수를 기술할 때 100L처럼 수 뒤에 L을 기술
- **파일에 내용을 쓰거나 읽으려면 그 파일 위치로 이동**
  - 파일 위치는 파일 내부에서 자료를 읽거나 쓰는 만큼 파일의 현재 위치에서 뒤로 이동

파일 스트림 연결 시 파일 위치

- **파일을 처음으로 열면 모드에 관계없이 파일 위치는 모두 0**
- **파일 모드가 추가(a)**
  - 파일을 처음 열면 파일 위치는 0이나 자료를 파일에 쓰면 자동으로 파일 위치가 마지막으로 이동되어 추가
  - 파일 위치를 임의로 이동하였다면 파일의 마지막으로 이동하여 추가

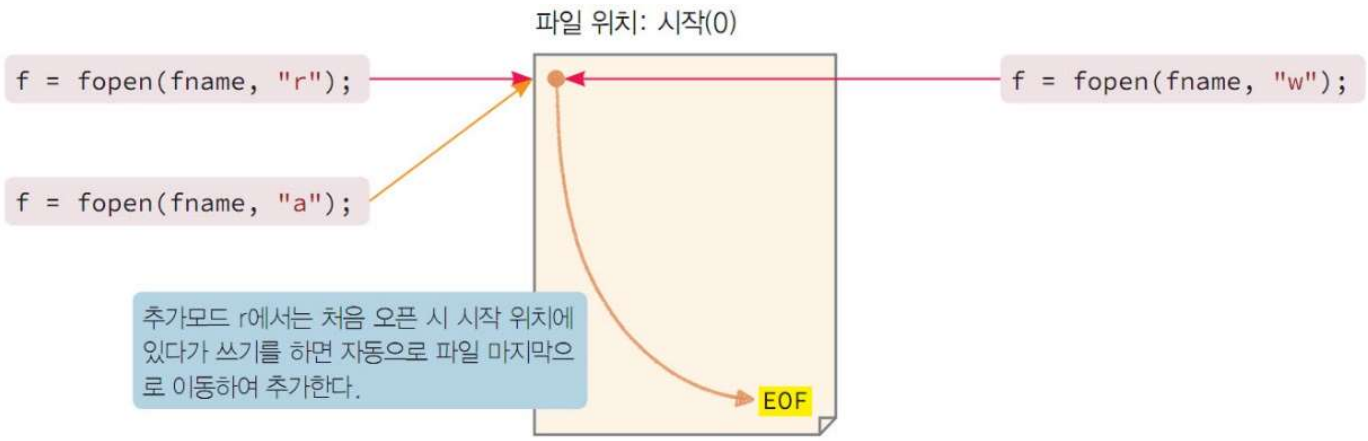


그림 15-27 파일 스트림 연결 시 처음 파일 위치

- **순차적 접근(sequential access)**
  - 파일 위치를 처음부터 하나씩 증가키면서 파일을 참조하는 방식
- **임의 접근(random access)**
  - 파일의 어느 위치든 바로 참조하는 방식
  - 관련 함수 fseek(), ftell(), rewind() 등을 활용하여 접근

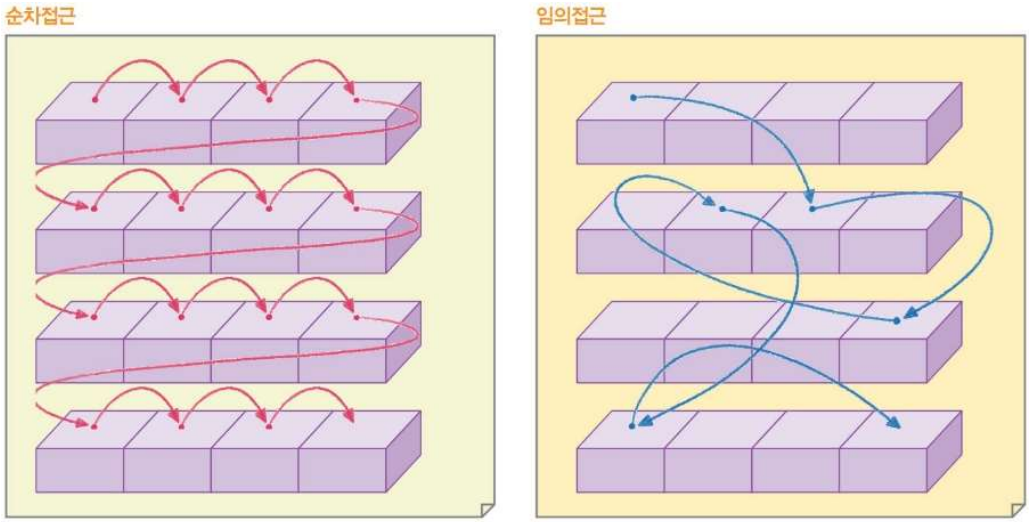


그림 15-28 순차적 접근과 임의 접근

함수 fseek(1)

- **파일의 임의 접근을 처리**
  - 함수 fseek()이 필요
    - 파일 위치를 자유자재로 이동
    - 헤더파일 stdio.h 필요

함수 fseek() 함수원형

```
int fseek(FILE * _File, long _Offset, int _Origin);
```

함수 fseek()는 파일 _File의 기준점 _Origin에서 _Offest만큼 파일 포인터를 이동하는 함수 , 성공하면 0을 반환하며 실패하면 0이 아닌 정수를 반환

```
fseek(f, 0L, SEEK_SET);
fseek(f, 100L, SEEK_CUR);
fseek(f, -100L, SEEK_END);
```

그림 15-29 함수 fseek() 함수원형

  - 첫번째 인자는 파일 포인터를 나타내며,
  - 두 번째 인자는 long 유형으로 기준점으로부터 떨어진 값을 말하며
    - 흔히 오프셋(offset)이라 부름
  - 세 번째 인자는 오프셋을 계산하는 기준점
    - 정수형 기호 상수로 다음 세 가지 중의 하나.

표 15-3 파일의 오프셋 기준의 종류를 나타내는 상수

기호	값	의미
SEEK_SET	0	파일의 시작 위치
SEEK_CUR	1	파일의 현재 위치
SEEK_END	2	파일의 끝 위치

함수 fseek(2)

- **함수 fseek(f, 100L, SEEK_SET)의 호출**
  - 파일 위치를 파일의 처음 위치에서 100바이트 떨어진 위치로 이동
- **함수 fseek(f, 100L, SEEK_CUR)의 호출**
  - 파일의 현재 위치에서 100바이트 떨어진 위치로 이동
- **함수 fseek(f, -100L, SEEK_END)의 호출**
  - 파일 끝 위치에서 앞으로 100바이트 떨어진 위치로 이동
- **함수 fseek()에서 offset**
  - 양수이면 파일의 끝점으로,
  - 음수이면 파일의 시작점으로의 이동방향을 표시

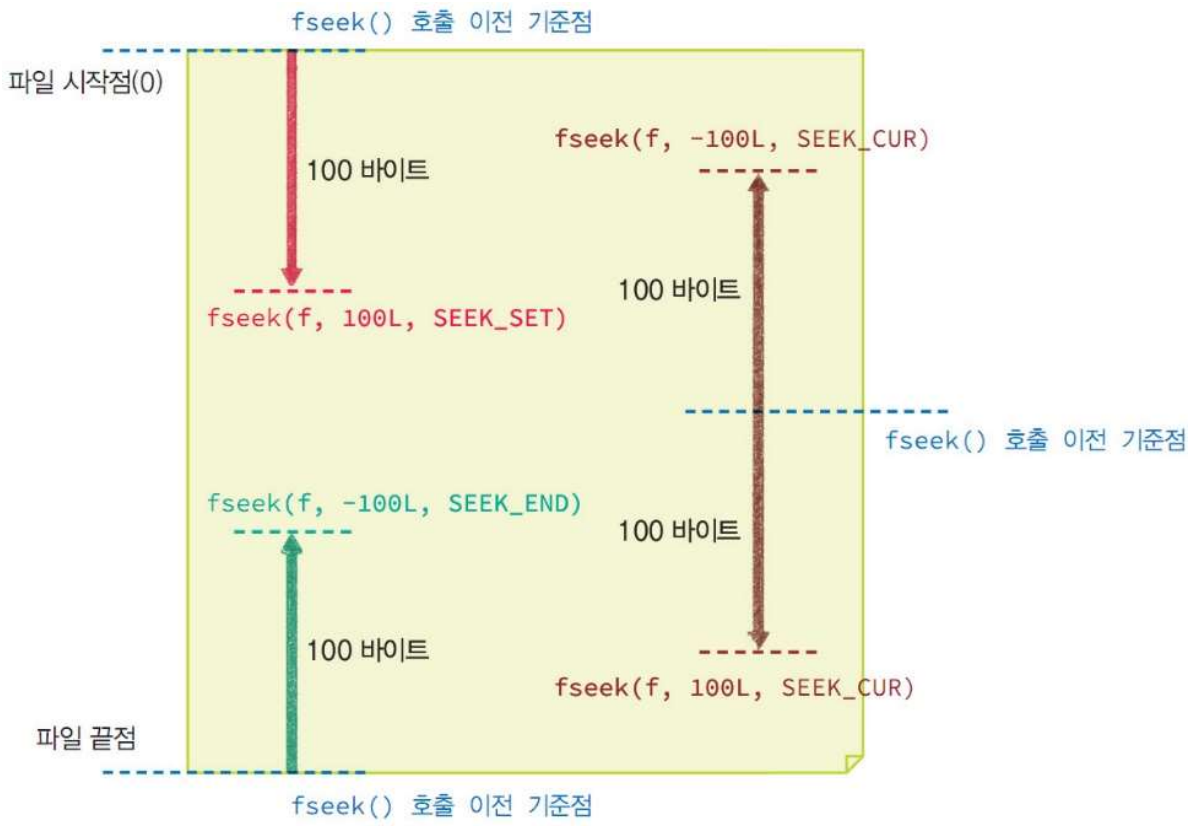


그림 15-30 함수 fseek()에서 여러 mode에 따른 offset의 의미

함수 fseek(3)

- **파일 위치와 관련된 함수**
  - 함수 ftell(): 인자인 파일의 파일 위치를 반환
  - 함수 rewind(): 파일 위치를 무조건 가장 앞으로 이동

표 15-4 파일 위치 관련 함수

함수	기능
`int fseek(FILE *, long offset, int pos)`	파일 위치를 세 기준점(pos)으로부터 오프셋(offset)만큼 이동
`long ftell(FILE *)`	파일의 현재 파일위치를 반환
`void rewind(FILE *)`	파일의 현재 위치를 0 위치(파일의 시작점)로 이동

- **함수 rewind() 호출: 파일의 맨 처음으로 이동**
  - 함수 fseek(f, 0L, SEEK_SET)

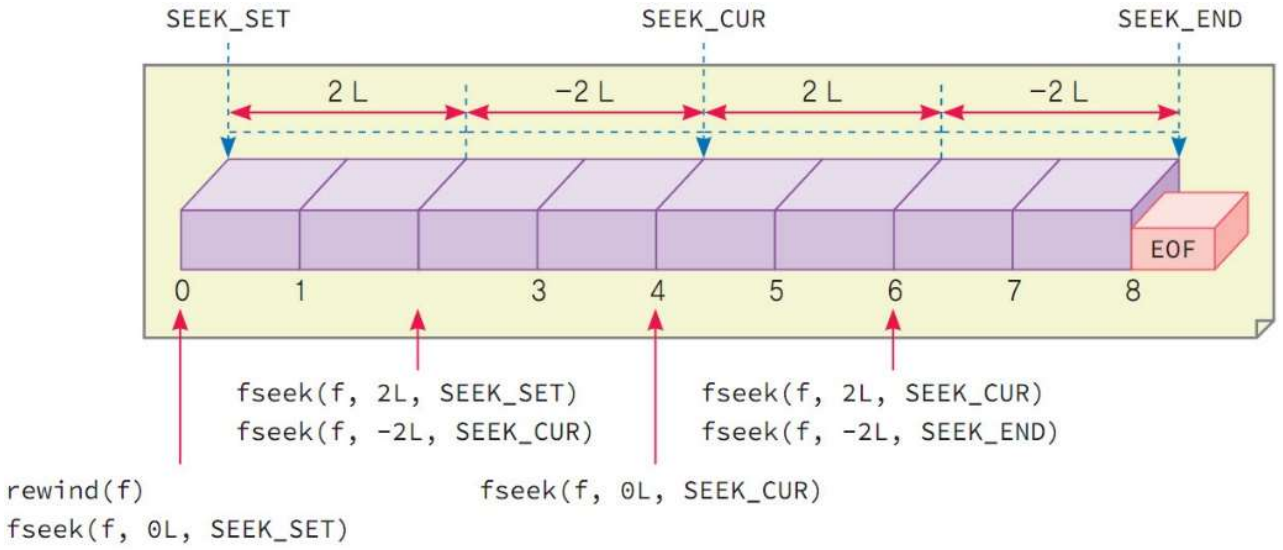


그림 15-31 파일 위치 이동 함수

파일 열기 다양한 모드(1)

- **함수 fopen()과 fopen_s()의 인자인 파일열기 종류(모드)**
  - 텍스트 파일인 경우 "r", "w", "a", "r+", "w+", "a+" 등

표 15-5 파일 열기 함수 fopen_s()의 모드 종류

모드	의미			
	파일 열기 모드	모드 전환	파일이 있는 경우	파일이 없는 경우
r	읽기(read)	쓰기(write) 불가능	파일의 처음에서 읽기 시작	에러 발생
w	쓰기(write)	읽기(read) 불가능	이전 내용이 지워지고 파일의 처음부터 쓰기 시작	새로 생성
a	추가(append)	읽기(read) 불가능	파일의 마지막에서 파일의 쓰기 시작하며, 파일 중간에 쓰는 것은 불가능	새로 생성
r+	읽기(read)	쓰기(write)	파일의 처음에서 읽기 시작	에러 발생
w+	쓰기(write)	읽기(read)	이전 내용이 지워지고 파일의 처음부터 쓰기 시작	새로 생성
a+	추가(append)	읽기(read)	파일의 마지막에서 파일의 쓰기 시작하며, 파일 중간에 쓰는 것은 불가능	새로 생성

- **파일모드**
  - 읽기모드 r
    - 읽기가 가능한 모드이며, 쓰기는 불가능
  - 쓰기모드 w
    - 파일 어디에든 쓰기가 가능한 모드이나 읽기는 불가능
  - 추가모드 a
    - 파일 중간에 쓸 수 없으며 파일 마지막에 추가적으로 쓰는 것만 가능한 모드
    - 읽기는 불가능
    - 파일에 쓰는 내용은 무조건 파일 마지막에 추가

파일 열기 다양한 모드(2)

- **파일모드에서 +의 삽입은 수정(update) 모드 의미**
  - 원래의 모드에서 읽기 또는 쓰기가 추가되는 모드
  - 수정(update) 모드에서는 모드 간의 전환이 가능
- **파일모드 r+**
  - 처음에 읽기 모드로 파일을 열어 쓰기 모드로 전환 가능
  - 파일이 없으면 오류가 발생
- **파일모드 w+**
  - 처음에 쓰기 모드로 파일을 열어 필요하면 읽기 모드로 전환 가능
  - 만일 파일이 존재한다면 이전의 내용은 모두 사라짐
- **파일모드 a+**
  - 처음에 추가 모드로 파일을 열어 필요하면 읽기 모드로 전환 가능
- **수정모드에서 모드전환**
  - 추가모드와 읽기모드 간, 쓰기모드와 읽기 모드간의 전환이 가능
  - 파일모드 전환 사이에는 fflush()와 fseek() 또는 rewind()와 같은 함수 호출이 필요
- **이진 파일을 위한 파일 열기 모드**
  - 문자 'b'를 추가 가능

그림 15-32 모드전환 시 필요한 함수 호출

표 15-6 이진파일 열기 함수 fopen()의 모드 종류

모드		의미
rb		이진파일의 읽기(read) 모드로 파일을 연다.
wb		이진파일의 쓰기(write) 모드로 파일을 연다.
ab		이진파일의 추가(append) 모드로 파일을 연다.
rb+	r+b	이진파일의 읽기(read)와 쓰기(write) 모드로 파일을 연다.
wb+	w+b	이진파일의 읽기(read)와 쓰기(write) 모드로 파일을 연다.
ab+	a+b	이진파일의 추가(append) 모드로 파일을 연다.

학생 성적 정보를 추가(1)

예제 appendscorefile.c

- 많은 학생의 성적정보를 추가하는 프로그램
- 실행되면 제일 먼저 파일 score.bin 파일에 있는 학생 정보를 모두 읽어와 출력
- 파일에 있는 마지막 학생 정보로부터 마지막 학생 번호를 알아냄
  - 이 번호에 1씩 증가시키면서 다음에 추가될 학생의 번호로 이용
- 추가될 학생 정보는 학생마다 한 행씩 자료를 받아서 파일 score.bin 파일에 추가

실습예제 15-8 appendscorefile.c

학생 성적 구조체 정보를 추가하는 프로그램

```c
// file: appendscorefile.c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

struct personscore
{
   int number;
   char name[40];
   int mid;
   int final;
   int quiz;
};
typedef struct personscore pscore;

void printhead();
int printscore(FILE *f);
void appendscore(FILE *f, int cnt);

int main()
{
   char fname[] = "score.bin";
   FILE *f;
   int cnt = 0;
   long offset = 0;

   if ((f = fopen(fname, "ab+")) == NULL)
   //if (fopen_s(&f, fname, "a+") != 0)
   {
      printf("파일이 열리지 않습니다.\n");
      exit(1);
   }
   int readcnt = printscore(f);
   if (readcnt == 1)
   {
      pscore score;
      offset = (long) sizeof(pscore); //구조체 하나의 크기
      //파일의 마지막에서 마지막 학생을 읽기 위해 한 학생만큼 뒤로 이동
      fseek(f, -offset, SEEK_END);
      //fseek(f, -offset, SEEK_CUR); //파일의 현재 포인터에서 한 학생만큼 뒤로 이동
      fread(&score, sizeof(pscore), 1, f); //마지막 학생을 읽음

      //제일 마지막 자료의 번호를 찾기 위하여
      cnt = score.number;
      printf("\n제일 마지막 번호가 %d번 입니다. \n\n", cnt);
   }
   fseek(f, 0L, SEEK_END);
   appendscore(f, cnt);
   printscore(f);
   fclose(f);

   return 0;
}

void appendscore(FILE *f, int cnt)
{
   pscore score = {0};
   char line[80];
   printf("추가할 이름과 성적(중간, 기말, 퀴즈)을 입력하세요.\n\n");
```

학생 성적 정보를 추가(2)

예제 appendscorefile.c

- 키보드 ctrl + z를 누르면 입력이 종료
  - 다시 파일 score.bin에서 모든 자료를 읽어 모든 정보를 출력
  - 프로그램에서 제일 먼저 자료를 추가할 score.bin 파일을 모드 "ab+"로 열기
  - 파일을 "ab+"로 여는 이유는 학생 정보를 추가도 하고, 다시 읽기도 하기 위함

```
61      fgets(line, 80, stdin);
62      while (!feof(stdin)) {
63          sscanf_s(line, "%s %d %d %d", score.name, 40, &score.mid, &score.
                final, &score.quiz);
64          score.number = ++cnt;
65          fwrite(&score, sizeof(pscore), 1, f);
66          fgets(line, 80, stdin);
67      }
68  }
69
70  int printscore(FILE *f)
71  {
72      //파일의 맨 앞으로 이동
73      rewind(f);
74      pscore score;
75      //파일 f에 하나도 자료가 없으면 변수 readcnt가 0
76      int readcnt = fread(&score, sizeof(pscore), 1, f);
77      if (readcnt == 0) {
78          printf("현재는 성적 정보가 하나도 없습니다. >>\n");
79          return 0;
80      }
81          //제목 출력
82      printhead();
83      while (!feof(f)) {
84          //표준출력에 쓰기
85          fprintf(stdout, "%6d%18s%8d%8d%8d\n",
86              score.number, score.name, score.mid, score.final, score.quiz);
87          fread(&score, sizeof(pscore), 1, f);
88      }
89      fprintf(stdout, "%s\n", "  ----------------------------------------");
90
91      return 1;
92  }
93
94  void printhead()
95  {
96      printf("\n현재의 성적 내용은 >>\n");
97      fprintf(stdout, "%s\n", "  ----------------------------------------");
98      fprintf(stdout, "%8s%15s%10s%8s%8s\n", "번호", "이름", "중간", "기말", "퀴즈");
99      fprintf(stdout, "%s\n", "  ----------------------------------------");
100 }
```

설명

```
23  읽거나 쓸 파일의 이름이 저장되는 변수
25  학생 번호가 저장될 변수
```

학생 성적 정보를 추가(3)

예제 appendscorefile.c

- 키보드 ctrl + z를 누르면 입력이 종료
  - 다시 파일 score.bin에서 모든 자료를 읽어 모든 정보를 출력
  - 프로그램에서 제일 먼저 자료를 추가할 score.bin 파일을 모드 "ab+"로 열기
  - 파일을 "ab+"로 여는 이유는 학생 정보를 추가도 하고, 다시 읽기도 하기 위함

```
26    구조체 pscore의 크기 저장
28    함수 fopen()의 호출 시, 첫 인자는 파일이름, 두 번째 인자는 모드로 "ab+"는 이진 추가모드
      이며, 반환값을 파일 포인터로 선언한 f에 대입, 만일 파일 열기에 실패하면 f에 NULL이 저장됨
31~32 파일 열기에 실패하면 메시지 출력하고 exit(1)으로 종료
34    파일에 이미 자료가 있으면 먼저 출력하기 위해 printscore() 호출
35    변수 readcnt가 1이면 이미 학생 정보가 있는 것을 의미
38    변수 offset에는 구조체 pscore의 크기를 저장
40    파일의 마지막에서 마지막 학생을 읽기 위해 한 학생만큼 뒤로 이동
42    마지막 학생을 읽어 변수 score에 저장
45    변수 cnt에 학생 번호를 저장
48    파일 f에서 마지막으로 이동
49    학생을 추가하기 위해 함수 appendscore() 호출
50    모든 학생의 정보를 다시 출력하기 위해 함수 printscore() 호출
51    파일 처리가 종료되었으면 fclose()로 스트림을 닫음
61    표준입력에서 한 행을 입력해 변수 line에 저장
62    파일 f가 EOF이 아니면 반복 계속
63    문자열 입력으로 구조체 pscore 자료인 score에 여러 멤버를 저장
65    다시 이진모드로 파일 f에 score 구조체를 저장 저장
66    다시 표준입력에서 한 행을 입력해 변수 line에 저장
73    파일의 맨 앞으로 이동
76    파일 f에서 구조체 하나를 읽어 score에 저장, 파일 f에 하나도 자료가 없으면 변수 readcnt가 0
79    파일 f에 하나도 자료가 없으면 0을 반환하고 종료
82    제목 출력
83    파일 f가 EOF가 아니면 반복 계속
85~86 표준출력으로 구조체 pscore 자료인 score를 출력
87    다시 이진모드로 파일 f에서 구조체 pscore 자료를 하나 읽어 변수 score에 저장
94~100  파일 f의 현재 정보를 모두 출력하는 함수
```

실행결과

```
현재의 성적 내용은 >>
-----------------------------------------------
  번호          이름      중간      기말      퀴즈
-----------------------------------------------
     1          안혜지      89        76        98
     2          조손정      76        87        59
     3          김혜원      98        89        99
-----------------------------------------------

제일 마지막 번호가 3번 입니다.

추가할 이름과 성적(중간, 기말, 퀴즈)을 입력하세요.

이미림 98 87 92
리디아고 78 76 98
^Z
```

```
현재의 성적 내용은 >>
-----------------------------------------------
  번호          이름      중간      기말      퀴즈
-----------------------------------------------
     1          안혜지      89        76        98
     2          조손정      76        87        59
     3          김혜원      98        89        99
     4          이미림      98        87        92
     5          리디아고    78        76        98
-----------------------------------------------
```

- **표준 입출력 장치를 이용한 입출력 함수**
  - 대부분 헤더파일 stdio.h 필요

표 15-7 자료에 따른 다양한 입출력 함수

자료	종류	표준 입출력	파일 입출력
문자	입력	`int getchar(void)`	`int getc(FILE *)` `int fgetc(FILE *)`
	출력	`int putchar(int)`	`int putc(int, FILE *)` `int fputc(int, FILE *)`
문자열	입력	`char * gets(char *)`	`char * fgets(char *, int, FILE *)`
	출력	`int puts(const char *)`	`int fputs(const char *, FILE *,)`
서식 자료	입력	`int scanf(const char *, ...)` `int scanf_s(const char *, ...)`	`int fscanf(FILE *, const char *, ...)` `int fscanf_s(FILE *, const char *, ...)`
	출력	`int printf(const char *, ...)`	`int fprintf(FILE *, const char *, ...)`

이진 파일

- **블록과 정수 자료의 파일 입출력 함수**
  - 함수 getw()와 putw()
    - 워드(word) 크기의 int 형 정수를 파일에 이진 모드로 입출력하는 함수
  - 헤더파일 stdio.h 필요

표 15-8 블록과 정수의 파일 입출력 함수

자료	종류	파일 입출력
블록	입력	`size_t fread(void *, size_t, size_t, FILE *)`
	출력	`size_t fwrite(const void *, size_t, size_t, FILE *)`
정수(int)	입력	`int getw(FILE *)` `int _getw(FILE *)`
	출력	`int putw(int, FILE *)` `int _putw(int, FILE *)`

파일 처리 함수 remove()와 rename()

- **함수 remove()**
  - 지정된 특정한 파일을 삭제
  - remove("sample.txt")
    - 문자열로 지정된 파일을 삭제
  - 파일이 성공적으로 삭제되면 0을 반환하며, 실패하면 -1을 반환
- **함수 rename()**
  - 지정된 파일 또는 폴더의 이름을 새로운 이름으로 바꾸는 기능을 수행
  - rename("oldname.txt", "newname.txt")
    - 앞의 파일이름 또는 폴더이름을 뒤에 지정한 이름으로 바꾸는 역할
  - 성공적으로 이름이 수정되면 0을 반환하며, 실패하면 0이 아닌 정수를 반환
- **헤더 파일 stdio.h에 그 함수 원형이 정의**

표 15-9 파일 삭제 함수

기능	함수 원형
파일 삭제	`int remove(const char *filename)`
파일 또는 폴더 이름 바꾸기	`int rename(const char *old, const char *new)`

파일 이름 수정

예제 rename.c

- 함수 rename()을 이용한 예제 프로그램
- 이 프로그램을 Visual C++에서 실행
  - 메뉴 [프로젝트/rename 속성]를 선택한 대화상자에서 [명령인수]에 화면에 수정하고 싶은 파일의 예전 이름과 새 이름을 기술
- 프로젝트 폴더에 파일 old.c를 하나 만들고 다음 rename 프로그램을 실행
  - 이제 프로젝트 폴더에서 파일 old.c가 new.c로 변경된 것을 확인

실습예제 15-9 rename.c

함수 rename()을 사용하여 파일 이름을 수정하는 프로그램

```
01  // file: rename.c
02  #include <stdio.h>
03  #include <stdlib.h>
04
05  int main(int argc, char *argv[])
06  {
07     if (argc < 3)
08     {
09        printf("사용법: rename from to");
10        exit(1);
11     };
12
13     //파일 이름 수정 함수 호출
14     rename(argv[1], argv[2]);
15     printf("파일 %s가 %s로 수정되었습니다.\n", argv[1], argv[2]);
16
17     return 0;
18  }
```

설명

14 인자 argv[1], argv[2]로 함수 rename() 호출
15 수정된 내용 출력

실행결과

```
파일 old.c가 new.c로 수정되었습니다.
```

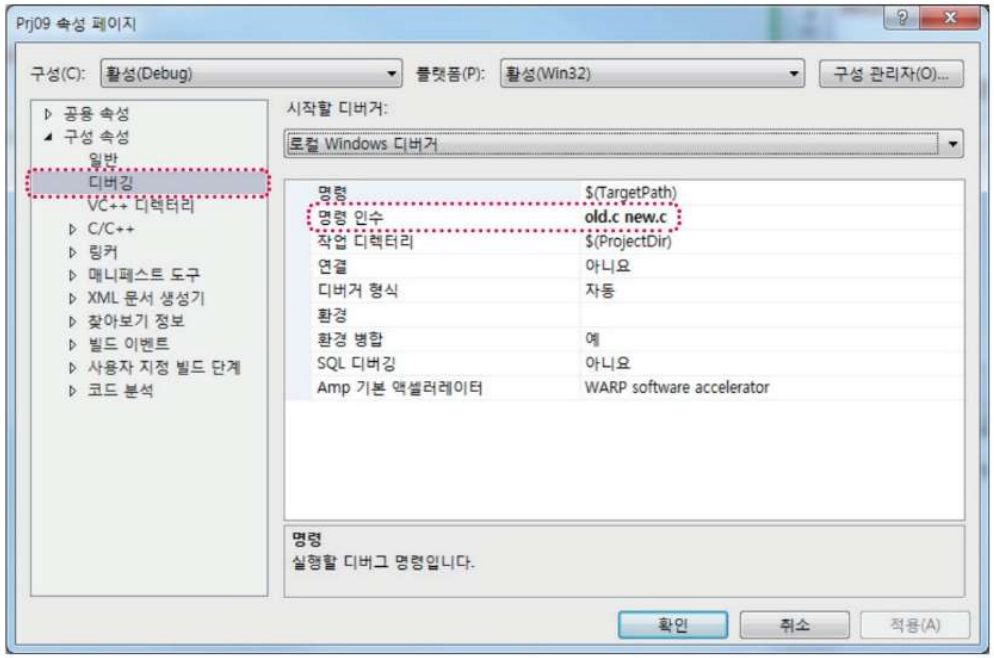


그림 15-33 Visual C++에서 프로그램 rename을 위한 명령행 인자

LAB 정수 배열값을 파일에 출력한 후 다시 입력(1)

- **일차원 정수 배열의 내용을 모두 파일에 출력한 후**
  - 다시 파일에서 입력으로 받아 표준 출력
- **방법**
  - 파일은 "wb+"로 열어 이진모드로 쓰고 읽을 수 있도록
  - 함수 _putw()를 사용해 출력하고 함수 _getw()를 사용해 입력
  - 배열의 내용을 파일 "test.bin"에 모두 출력
  - 파일에 출력 후, 다시 처음부터 읽으려면 함수 rewind()를 호출
  - 파일 "test.bin"에서 정수를 읽을 때는 한 정수 읽은 후 함수 fseek()을 사용하여 파일 위치를 하나 건너 뛰고 다음 정수를 읽는 식으로 입력하여 다시 표준출력
- **결과**
  - 파일에 출력 자료: 10 20 30 40 50 60 70 80
  - 파일에서 입력 자료(하나씩 건너 뜀): 10 30 50 70

LAB 정수 배열값을 파일에 출력한 후 다시 입력(2)

Lab 15-4 arrayput.c

```c
//file: arrayput.c
#include <stdio.h>
#include <stdlib.h>

int main()
{
   FILE *f;
   if (fopen_s(&f, "test.bin", "wb+") != 0)
   //if ((f = fopen("test.bin", "wb+")) == NULL)
   {
      printf("파일이 열리지 않습니다.\n");
      exit(1);
   }

   int out[] = { 10, 20, 30, 40, 50, 60, 70, 80 };
   //_putw() 이용 배열 out의 내용 모두 출력
   int size = sizeof(out) / sizeof(out[0]);
   printf("파일에 출력 자료: ");
   for (int i = 0; i < size; i++) {
      _putw(out[i], f);
      printf("%d ", out[i]);
   }
   printf("\n");

   //파일 f의 파일 지시자를 0으로 위치
   ____________________;
   printf("파일에서 입력 자료(하나씩 건너 뜀): ");
   for (int i = 0; i < size / 2; i++)
   {
      int _________________________;
      fseek(_____________________________);
      printf("%d ", in);
   }
   printf("\n");
   fclose(f);

   return 0;
}
```

정답

```c
rewind(f);
   int in = _getw(f);
   fseek(f, sizeof(int), SEEK_CUR);
```

Thank you