11장 입력과 출력

getchar()/putchar()

- 함수 원형

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

- getchar()
  - 표준 입력 장치로 문자를 하나 읽는 함수
  - 읽을 문자가 없으면 EOF 리턴
- putchar()
  - 표준 출력 장치로 문자를 하나 쓰는 함수

표준 입출력 장치

- 표준 입력 장치 : 디폴트로는 키보드
- 표준 출력 장치 : 디폴트로는 화면
- 표준 입출력 장치는 변경할 수 있음
  - 입출력 재지정
  - 파이프
  - …

표준 입출력 장치

- 입출력 재지정(<, >)

  $ io_prog < in_file > out_file

  - io_prog의 표준 입력 장치는 in_file 파일이되고,
  - 표준 출력 장치는 out_file 파일이 됨

- 파이프

  $ io_prog1 | io_prog2 | io_prog3

  - io_prog2의 표준 입력 장치는 첫 번째 파이프 (io_prog1의 표준 출력)
  - io_prog2의 표준 출력 장치는 두 번째 파이프 (io_prog3의 표준 입력)

getchar()/putchar()

프로그램 11.1

```
#include <stdio.h>
int main(void)
{
    int   c;
    while ((c = getchar()) != EOF) {
        putchar(c);
    }

    return 0;
}
```

프로그램 결과

```
$ pass
abcdefg
abcdefg
hij
hij
^d
$ ls | pass
bin
etc
include
lib
local
man
sbin
tmp
var
$ pass < infile > outfile
```

printf()

- 첫 번째 인자를 제어 문자열이라고 함
- 제어문자열에서 %부터 변환 문자까지를 변환 명세라고 함
- 첫 번째 인자인 제어 문자열을 화면에 출력하는 데, 변환 명세가 나오면 뒤의 인자를 적절히 형 변환하여 대신 출력 함

```
printf("%s 학생의 점수는 %d 점입니다.",name,grade);
```

- 변환 명세보다 인자가 많으면 여분의 인자는 무시됨
- 인자가 적으면 시스템 종속적인 일이 일어남

printf()

- **변환 명세의 형식**

  %[플래그][폭][.정밀도][형변환자]변환문자

  - 플래그, 폭, .정밀도, 형변환자 : 옵션
  - 변환문자 : 생략할 수 없음

printf()

변환문자	출력 형태
c	문자
d, i	10진 정수
u	부호 없는 10진 정수
o	부호 없는 8진 정수
x, X	부호 없는 16진 정수, 예: 5dee, 5DEE
f, F	부동 소수점 표기법의 실수, 예: 7.123000
e, E	지수 표기법의 실수, 예: 7.123000e+00, 7.123000E+00
a, A (C99)	16진 지수 표기법의 실수, 예: 0x7.123p+20, 0X7.123P+20
g, G	e형식과 f 형식 중 짧은 것 또는 E형식과 F 형식 중 짧은 것
s	널로 끝나는 문자열
p	포인터 값의 16진 정수
n	출력되는 것 없음, 대응되는 정수형 포인터 인자에 현재까지 출력된 문자의 개수를 배정함
%	% 문자, 대응되는 인자 없음

printf()

- 플래그
  - 출력 형식 조정

플래그	의미
-	필드에서 좌측 정렬로 출력
+	숫자 앞에 +나 -를 항상 붙여 출력
공백	양수 앞에 공백을 붙여 출력
0	숫자를 우측 정렬로 출력할 때 남는 공간을 0으로 채워서 출력
#	8진수는 앞에 0, 16진수는 앞에 0x를 출력

* 필드 : 인자가 출력되는 공간(자리)

printf()

- 폭
  - 필드의 크기(자리수) 지정
  - 양의 정수나 *
    - * : 인자로 받아들임
  - 예

```
printf("i = %8d\n", i);
printf("i = %*d\n", 8, i);
```

printf()

- 정밀도
  - 점(.) 뒤에 명시
  - 음이 아닌 정수나 *
    - * : 인자로 받아들임
- **정밀도는 변환 문자 별로 의미가 다름**
  - a, A, e, E, f, F : 소수점 이하의 자릿수
  - g, G : 최대 유효 숫자
  - s : 문자열로부터 출력될 문자의 최대 개수

printf()

- 형변환자
  - 인자의 크기 지정

형변환자	뒤에 올 수 있는 변환 문자	변환 형
hh	d, i, o, u, x, X	signed/unsigned char
h	d, i, o, u, x, X	signed/unsigned short
l	d, i, o, u, x, X	signed/unsigned long
ll	d, i, o, u, x, X	signed/unsigned long long
z	d, i, o, u, x, X	size_t
L	a, A, e, E, f, F, g, G	long double

printf()

```
printf("%u", 256);                    256
printf("%hhu", 256);                  0
printf("%d", 2100000000);             2100000000
printf("%d", 2100000000001);          -474836479
```

warning: integer constant is too large for "long" type

```
printf("%d", 2100000000001LL);        -474836479
printf("%lld", 2100000000001LL);      2100000001
```

printf()

프로그램 11.2

```
#include <stdio.h>
int main(void){
    char c = 'A', s[] = "test string";
    int   i = 1024;
    int   j = 9, k = 2;
    long  l = 12345678;
    long long ll = 12345678901234LL;
    float f = 123.45678, f0 = 987.0;
    double d = -123.45678, d0 = 0.00000009;

    printf(">>문자 출력<<\n");
    printf("%%c: |%c|, %%7c: |%7c|,  %%+7c: |%+7c|\n", c, c, c);
    printf("%%s: |%s|, %%7s: |%7s|,  %%+7s: |%+7s|\n", s, s, s);
    printf("%%.7s: |%.7s|, %%10.7s: |%10.7s|,  %%+10.7s: |%+10.7s|\n",
           s, s, s);
    printf("C는 재미있는 %n\n", &j);
    printf("%*c프로그래밍 언어입니다.\n", j, ' ');
```

프로그램 결과

```
>>문자 출력<<
%c: |A|, %7c: |      A|,  %+7c: |      A|
%s: |test string|, %7s: |test string|,  %+7s: |test string|
%.7s: |test st|, %10.7s: |   test st|,  %+10.7s: |   test st|
C는 재미있는
                프로그래밍 언어입니다.
```

프로그램 11.2

```
int   i = 1024;
int   j = 9, k = 2;
long  l = 12345678;
long long ll = 12345678901234LL;

printf("\n>>정수 출력<<\n");
printf("%%d: |%d|, %%o: |%o|, %%x: |%x|,  %%u: |%u|\n", i, i, i, i);
printf("%%07d: |%07d|, %%0.7d: |%0.7d|, %%+d: |%+d|\n", i, i, i);
printf("%%X: |%X|, %%#x: |%#x|, %%#X: |%#X|\n", i, i, i);
printf("%%ld: |%ld|, %%lo: |%lo|, %%lx: |%lx|\n", l, l, l);
printf("%%lld: |%lld|, %%llo: |%llo|, %%llx: |%llx|\n", ll, ll, ll);
```

프로그램 결과

```
>>정수 출력<<
%d: |1024|, %o: |2000|, %x: |400|,  %u: |1024|
%07d: |0001024|, %0.7d: |0001024|, %+d: |+1024|
%X: |400|, %#x: |0x400|, %#X: |0X400|
%ld: |12345678|, %lo: |57060516|, %lx: |bc614e|
%lld: |12345678901234|, %llo: |263516363427762|, %llx: |b3a73ce2ff2|
```

프로그램 11.2

```
    float f = 123.45678, f0 = 987.0;
    double d = -123.45678, d0 = 0.00000009;

    printf("\n>>실수 출력<<\n");
    printf("%%f: |%f|, %%e: |%e|,  %%g: |%g|\n", f, f, f);
    printf("%%.3f: |%.3f|, %%.3e: |%.3e|,  %%.3g: |%.3g|\n", f, f, f);
    printf("%%10.3f: |%10.3f|, %%10.3e: |%10.3e|,  %%10.3g: |%10.3g|\n",
           f, f, f);
    printf("%%*.*f: |%*.*f|\n", j, k, f);
    printf("%%f: |%f|, %%e: |%e|,  %%g: |%g|\n", f0, f0, f0);
    printf("%%#f: |%#f|, %%#e: |%#e|,  %%#g: |%#g|\n", f0, f0, f0);
    printf("%%f: |%f|, %%e: |%e|,  %%g: |%g|\n", d, d, d);
    printf("%%*f: |%*f|, %%.*f: |%.*f|,  %%*.*f: |%*.*f|\n",
           j, d, k, d, j, k, d);
    printf("%%f: |%f|, %%e: |%e|,  %%g: |%g|\n", d0, d0, d0);
    return 0;
}
```

프로그램 결과

```
>>실수 출력<<
%f: |123.456779|, %e: |1.234568e+02|,  %g: |123.457|
%.3f: |123.457|, %.3e: |1.235e+02|,  %.3g: |123|
%10.3f: |   123.457|, %10.3e: | 1.235e+02|,  %10.3g: |       123|
%*.*f: |       123.46|
%f: |987.000000|, %e: |9.870000e+02|,  %g: |987|
%#f: |987.000000|, %#e: |9.870000e+02|,  %#g: |987.000|
%f: |-123.456780|, %e: |-1.234568e+02|,  %g: |-123.457|
%*f: |-123.456780|, %.*f: |-123.46|,  %*.*f: |      -123.46|
%f: |0.000000|, %e: |9.000000e-08|,  %g: |9e-08|
```

scanf()

- 첫 번째 인자를 제어 문자열이라고 함
- 제어문자열에서 %부터 변환 문자까지를 변환 명세라고 함
- 입력 스트림에서 변환 명세 대로 읽어서 대응 인자에 배정함
  - 대응 인자는 포인터이어야 함
- 첫 번째 인자인 제어 문자열에서 제어 문자가 아닌 일반 문자는 입력 스트림에서 똑같은 문자를 제거함

```
scanf("name : %s", s);
```

scanf()

- **변환 명세의 형식**

  %[*][폭][형변환자]변환문자

  - *, 폭, 형변환자 : 옵션
  - 변환문자 : 생략할 수 없음

scanf()

변환문자	입력 형태	대응 인자 형
c	공백을 포함한 모든 문자	char 포인터
d, i	10진 정수 (부호는 옵션)	정수 포인터
u	10진 정수 (부호는 옵션)	부호없는 정수 포인터
o	8진 정수 (부호는 옵션)	부호없는 정수 포인터
x	16진 정수 (부호는 옵션)	부호없는 정수 포인터
a, e, f, g	실수 (부호는 옵션)	실수 포인터
s	공백 없는 문자열	char 포인터
p	보통 16진 정수 (시스템에 따라 다름)	void 포인터
n	지금까지 읽은 문자 개수를 대응 인자에 배정, 입력 스트림의 내용은 안 읽음	정수 포인터
%	입력 스트림에서 % 읽음	대응 인자 없음
[ ]	다음에 설명	char 포인터

scanf()

- 변환문자 [ ]
  - 원하는 문자들(스캔집합)로만 구성된 문자열을 읽어들임
  - [ ] 내의 첫 번째 문자가 ^가 아니면 괄호 내의 문자가 스캔집합이 되고, ^이면 괄호 내의 문자를 제외한 문자들이 스캔집합이 됨
  - 입력 스트림에서 스캔집합 이외의 문자가 나올 때까지 읽어서 대응 인자에 배정함

```
scanf("%[^ \n\t]", s);   // 공백, 개행, 탭 제외
scanf("%[0-9a-fA-F]", s); // 숫자와 알파벳
```

scanf()

- *
  - 입력 스트림의 내용을 지움
- **예제**

```
scanf("%d %*d %d", &a, &b);
```

  - 입력 스트림 : 20 40 50
  - 20은 a에, 40은 무시, 50은 b에 저장

scanf()

- 폭
  - 읽어 들일 필드의 최대 크기(문자 수) 지정
- **예제**

```
scanf("%3d %5c", &a, s);
```

  - 입력 스트림 : 1234567890
  - 123은 a에, 45678은 s에 저장

scanf()

- 형변환자
  - 인자의 크기 지정

형변환자	뒤에 올 수 있는 변환 문자	인자의 형
hh	d, i, o, u, x, X, n	signed/unsigned char 포인터
h	d, i, o, u, x, X, n	signed/unsigned short 포인터
l	d, i, o, u, x, X, n	signed/unsigned long 포인터
l	a, A, e, E, f, F, g, G	double 포인터
ll	d, i, o, u, x, X, n	signed/unsigned long long 포인터
z	d, i, o, u, x, X, n	size_t 포인터
L	a, A, e, E, f, F, g, G	long double 포인터

scanf()

```
float f; double d;
scanf("%f", &f); //12.34를 입력한다면
printf("%f", f);                                    12.340000
scanf("%f", &d); //12.34를 입력한다면
printf("%f", d);                                    0.000000
scanf("%lf", &d); //12.34를 입력한다면
printf("%f", d);                                    12.340000
```

sprintf()/sscanf()

- printf()와 scanf()의 문자열 버전
- 표준 입출력 장치로 입출력하는 것이 아니라 문자열에 쓰거나 읽음
- 첫 번째 인자로 입출력을 위한 문자열이 오고 나머지 인자는 printf(), scanf()와 같음

sprintf()/sscanf()

프로그램 11.3

```
void cal(char * express, char *result){
   int   opd1, opd2;
   char  op;
   if (sscanf(express, "%d %c %d", &opd1, &op, &opd2) != 3) {
       sprintf(result, "수식 오류");
       return;
   }
   if (op == '+')
      sprintf(result, "%d", opd1 + opd2);
   else if (op == '-')
      sprintf(result, "%d", opd1 - opd2);
   else if (op == '*')
      sprintf(result, "%d", opd1 * opd2);
   else if (op == '/')
      sprintf(result, "%.3f", (float)opd1 / opd2);
   else
      sprintf(result, "수식 오류");
}
```

sprintf()/sscanf()

프로그램 11.3

```
int main(void){
    char express[21];
    char result[10];

    printf("수식을 입력 하세요 : ");
    scanf("%20[^\n]", express);
    cal(express, result);
    printf("%s = %s\n", express, result);

    return 0;
}
```

프로그램 결과

```
$ calculate
수식을 입력 하세요 : 19-3
19-3 = 16
$ calculate
수식을 입력 하세요 : 57 / 9
57 / 9 = 6.333
$ calculate
수식을 입력하세요 : 45* 543
45* 543 = 24435
$ calculate
수식을 입력 하세요 : 90   34
90   34 = 수식 오류
```

sprintf()/sscanf()

- sprintf()나 sscanf()는 호출될 때 마다 문자열의 처음부터 쓰거나 읽음

```
char str[] = "1234567890";
int a, b, c, d;
sscanf(str, "%2d%2d", &a, &b);
sscanf(str, "%2d%2d", &c, &d);
// a = 12, b = 34
// c = 12, d = 34
```

파일 입출력

- 파일 입출력을 위해서는 파일을 먼저 열고 해야함
- 관련 함수들

```
fopen()
fclose()
fprintf()
fscanf()
```

fopen()

- 파일을 열기 전에는 파일의 내용을 보거나 쓸 수 없음
- fopen() 함수는 파일 이름과 모드를 인자로 받음
- fopen()은 지정된 모드로 파일을 열고 그 파일을 접근할 수 있게 FILE 포인터를 리턴함
- 파일 열기를 실패하면 NULL을 리턴함

fopen()

- 모드

모드	의미
"r"	읽기 위해 문서 파일 열기
"w"	쓰기 위해 문서 파일 열기
"a"	첨부하기 위해 문서 파일 열기
"rb"	읽기 위해 이진 파일 열기
"wb"	쓰기 위해 이진 파일 열기
"ab"	첨부하기 위해 이진 파일 열기

  - 모드 뒤의 +는 파일을 읽기와 쓰기로 모두 연다는 것을 의미함

fopen()

- 사용 예

```
FILE *ifp, *ofp;
ifp = fopen("infile", "r");
ofp = fopen("outfile", "w");
```

  - infile과 outfile로 데이터를 읽고 쓸 때 ifp와 ofp를 통해 읽거나 써야 함

```
if ((ifp = fopen("infile", "r")) == NULL)    //오류 처리
   printf("오류 : 파일을 열 수 없습니다.");
else
   . . .              // 파일 읽기
```

표준 파일 포인터

- <stdio.h>에 정의

파일 포인터	설명	비고
stdin	표준 입력 파일	키보드로 연결됨
stdout	표준 출력 파일	화면으로 연결됨
stderr	표준 에러 파일	화면으로 연결됨

fclose()

- 파일을 다 사용한 후에는 fclose()를 사용하여 닫아야 함
- fclose()의 인자로는 fopen()에 의해 리턴된 FILE 포인터를 명시해야 함
- 예제

```
fclose(ifp);
fclose(ofp);
```

getc()/putc()

- FILE 포인터를 갖는다는 것만 제외하면 getchar()/putchar()와 같음
  - getchar() : getc(stdin)
  - putchar(c) : putc(c, stdout)
- 예제

```
c = getc(ifp);
putc(c, ofp);
```

getc()/putc()

프로그램 11.4

```
int main(int argc, char **argv){
   FILE  *ifp, *ofp;
   int    c;
   if (argc != 3) {
      printf("실행 오류 : \n   사용법 : %s from_file to_file\n", argv[0]);
      exit(1);
   }
   if ((ifp = fopen(argv[1], "r")) == NULL){
      printf("오류 : %s 파일을 열 수 없습니다.\n", argv[1]); exit(1);
   }
   if ((ofp = fopen(argv[2], "w")) == NULL){
      printf("오류 : %s 파일을 열 수 없습니다.\n", argv[2]); exit(1);
   }
   while ((c = getc(ifp)) != EOF)
      putc(c, ofp);
   fclose(ifp);
   fclose(ofp);
   return 0;
}
```

fprintf()/fscanf()

- printf()/scanf() 함수의 파일 버전
- 첫 번째 인자는 FILE 포인터이고 나머지는 printf()/scanf()와 같음

```
printf(...) : fprintf(stdout,...)
scanf(...) : fscanf(stdin, ...)
```

fprintf()/fscanf()

프로그램 11.5

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv)
{
   FILE  *pro, *sol;
   int    c, i, opd1, opd2;
   char  word[50];

   if (argc != 3) {
      fprintf(stderr, "실행 오류 :\n  사용법 : %s data_file out_file\n",
   argv[0]);
      exit(1);
   }
   if ((pro = fopen(argv[1], "w")) == NULL){
      fprintf(stderr, "오류 : %s 파일을 열 수 없습니다.\n", argv[1]);
      exit(1);
   }
```

fprintf()/fscanf()

프로그램 11.5

```
    for (i = 0; i < 50; i++)
        fprintf(pro, "%d + %d = \n", rand() % 1000, rand() % 1000);
    fclose(pro);
    if ((pro = fopen(argv[1], "r")) == NULL){
        fprintf(stderr, "오류 : %s 파일을 열 수 없습니다.\n", argv[1]);
        exit(1);
    }
    if ((sol = fopen(argv[2], "w")) == NULL){
        fprintf(stderr, "오류 : %s 파일을 열 수 없습니다.\n", argv[2]);
        exit(1);
    }

    while (fscanf(pro, "%d + %d = \n", &opd1, &opd2) != EOF)
        fprintf(sol, "%d + %d = %d\n", opd1, opd2, opd1 + opd2);

    fclose(pro);
    fclose(sol);

    return 0;
}
```

파일의 임의의 위치 접근

- 입출력 함수는 이전에 마지막으로 입출력이 일어난 곳부터 입출력을 실행함
  - 파일 위치 지시자
- 파일 위치 지시자와 관련된 함수들
  - ftell()
  - fseek()
  - rewind()

파일의 임의의 위치 접근

- ftell()
  - 파일 위치 지시자의 현재 값을 리턴
- 사용 예

```
pos = ftell(FILE_ptr);
```

  - FILE_ptr과 관련된 파일 위치 지시자를 pos에 배정
  - 리턴된 값은 파일의 처음부터 몇 바이트 떨어진 곳인가를 나타냄

파일의 임의의 위치 접근

- **fseek()**
  - 파일 위치 지시자의 값을 직접 지정함
- **사용 예**

```
fseek(FILE_ptr, offset, place)
```

  - 파일 위치 지시자를 place부터 offset 바이트 떨어진 곳을 나타내는 값으로 설정함
  - place

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

파일의 임의의 위치 접근

- **rewind()**
  - 파일 위치 지시자를 파일의 제일 앞으로 지정함
- **사용 예**

```
rewind(FILE_ptr)
```

  - fseek(FILE_ptr, 0, SEEK_SET)과 같음

파일의 임의의 위치 접근

프로그램 11.6

```
#include <stdio.h>
#define   MAXSTRING   100
int main(void){
   char   fname[MAXSTRING];
   int    c;
   FILE   *ifp;
   fprintf(stderr, "\n입력 파일 : ");
   scanf("%s", fname);
   ifp = fopen(fname, "r");
   fseek(ifp, 0, SEEK_END);
   if(ftell(ifp) == 0)
      return 0;
   fseek(ifp, -1, SEEK_CUR);
   while (1) {
         c = getc(ifp);
         putchar(c);
         if(ftell(ifp) == 1)
            break;
         fseek(ifp, -2, SEEK_CUR);
      }
      fclose(ifp);
      return 0;
   }
```

텍스트 파일

- putc()나 fprintf()는 문자로 출력함
  - 아스키 값으로 파일에 저장됨
  - 문서 편집기로 내용을 확인할 수 있음
- 텍스트로 저장하면 내용을 쉽게 파악할 수 있어 좋지만 파일의 크기가 커지고, 데이터의 직접 접근이 어려워짐

텍스트 파일

- 예

```
struct student{
   int   id;
   char  name[10];
   int   grade[3];  // 국어, 수학, 영어 성적
   int   sum;       // 성적 합
   float avg;       // 성적 평균
};
for (i = 0; i < N; i++)
   fprintf(output, "%d %s %d %d %d %d %f\n", st[i].id,
        st[i].name, st[i].grade[0], st[i].grade[1],
        st[i].grade[2], st[i].sum, st[i].avg);
```

  - 각 학생별 레코드 크기가 다름
  - 100번째 학생의 성적을 찾기 위해서는 처음부터 읽어야 함

이진 파일

- 메모리 내용과 같은 형식으로 작성된 파일
- 관련 함수
  - fwrite()
  - fread()

fwrite()

- 함수 원형

```
size_t fwrite(const void *buffer, size_t size,
                    size_t count, FILE *FP);
```

  - buffer : 파일에 쓸 데이터를 가지고 있는 포인터
  - size : 저장할 각 객체의 크기
  - count : 저장할 객체의 수
  - FP : 저장할 파일 포인터

fwrite()

프로그램 11.7

```
#include <stdio.h>
typedef struct student{
   int   id;
   char  name[10];
   int   grade[3];  // 국어, 수학, 영어 성적
   int   sum;       // 성적 합
   float avg;       // 성적 평균
} student;
int main(int argc, char **argv){
   FILE  *ofp;
   int    id, check;
   student  st = {0, "", {0}, 0, 0.0};
   if (argc != 2) {
      fprintf(stderr, "실행 오류 : \n   사용법 : %s out_file\n", argv[0]);
      exit(1);
   }
   if ((ofp = fopen(argv[1], "wb")) == NULL){
      fprintf(stderr, "오류 : %s 파일을 열 수 없습니다.\n", argv[1]);
      exit(1);
   }
```

프로그램 11.7

```
    fprintf(stderr, "성적을 입력하세요.\n");
    fprintf(stderr, "입력 형식 : 이름 국어성적 수학성적 영어성적\n");

    id = 1;
    check = scanf("%s %d %d %d", st.name,
                    &st.grade[0], &st.grade[1], &st.grade[2]);
    while (check != EOF)
    {
        st.id = id++;
        fwrite(&st, sizeof(student), 1, ofp);
        check = scanf("%s %d %d %d", st.name,
                    &st.grade[0], &st.grade[1], &st.grade[2]);
    }
    fclose(ofp);
    return 0;
}
```

프로그램 결과

```
$ input_grade output
성적을 입력하세요.
입력 형식 : 이름 국어성적 수학성적 영어성적
하나 78 90 100
둘 8 98 67
셋 78 90 88
넷 78 88 90
다섯 58 88 91
여섯 77 70 76
^D
$ cat output
□ 하나 N Z d
□ 둘 찮 Z d
□ 셋 찮 Z d
□ 넷 찮 Z d
□ 다섯 N Z d
□ 여섯 N Z d
```

fread()

- 함수 원형

```
size_t fread(const void *buffer, size_t size,
                    size_t count, FILE *FP);
```

  - buffer : 파일에서 읽은 데이터를 저장할 포인터
  - size : 각 객체의 크기
  - count : 객체의 수
  - FP : 읽을 파일 포인터

fread()

프로그램 11.8

```
#include <stdio.h>
typedef struct student{
   int   id;
   char  name[10];
   int   grade[3];  // 국어, 수학, 영어 성적
   int   sum;       // 성적 합
   float avg;       // 성적 평균
} student;
int main(int argc, char **argv){
   FILE  *ifp;
   int    check;
   student  st = {0, "", {0}, 0, 0.0};
   if (argc != 2) {
      fprintf(stderr, "실행 오류 : \n   사용법 : %s in_file\n", argv[0]);
      exit(1);
   }
   if ((ifp = fopen(argv[1], "rb")) == NULL){
      fprintf(stderr, "오류 : %s 파일을 열 수 없습니다.\n", argv[1]);
      exit(1);
   }
```

fread()

프로그램 11.8

```
    fprintf(stderr, "번호 이름 국어 수학 영어\n");

    check = fread(&st, sizeof(student), 1, ifp);
    while (check)
    {
        printf("%3d  %-5s %3d  %3d  %3d\n", st.id, st.name,
                    st.grade[0], st.grade[1], st.grade[2]);
        check = fread(&st, sizeof(student), 1, ifp);
    }
    fclose(ifp);
    return 0;
}
```

프로그램 결과

```
$ print_grade output
번호 이름  국어  수학 영어
   1  하나   78   90  100
   2  둘      8   98   67
   3  셋     78   90   88
   4  넷     78   88   90
   5  다섯   58   88   91
   6  여섯   77   70   76
   7  일곱  100   90    9
   8  여덟   98  100   87
   9  아홉   93   95   50
  10  열     66   90   88
```

이진 파일

프로그램 11.9 일부

```
case 1 :          // 전체 성적 출력
   rewind(fp);
   fprintf(stderr, "번호 이름 국어 수학 영어 총점 평균\n");
   check = fread(&st, sizeof(student), 1, fp);
   . . .
case 2 :          // 개인 성적 출력
   fprintf(stderr, "학생 id : ");
   scanf("%d", &id);
   fseek(fp, sizeof(student) * (id - 1), SEEK_SET);
   check = fread(&st, sizeof(student), 1, fp);
   . . .
case 3 :          // 성적 추가
   . . .
   fseek(fp, 0, SEEK_END);
   st.id = ftell(fp) / sizeof(student) + 1;
   . . .
   fwrite(&st, sizeof(student), 1, fp);
   fflush(fp);
```

이진 파일

프로그램 11.9 일부

```
case 4 :          // 성적 수정
   fprintf(stderr, "학생 id : ");
   scanf("%d", &id);
   fseek(fp, sizeof(student) * (id - 1), SEEK_SET);
   check = fread(&st, sizeof(student), 1, fp);
   . . .
   fseek(fp, -sizeof(student), SEEK_CUR);
   fwrite(&st, sizeof(student), 1, fp);
   . . .
```

프로그램 11.9 일부

```
case 5 :          // 성적 처리
   rewind(fp);
   check = fread(&st, sizeof(student), 1, fp);
   while (check){
      st.sum = st.grade[0] + st.grade[1] + st.grade[2];
      st.avg = st.sum / 3.0;
      fseek(fp, -sizeof(student), SEEK_CUR);
      fwrite(&st, sizeof(student), 1, fp);
      check = fread(&st, sizeof(student), 1, fp);
   }
   . . .
```

프로그램 결과

```
원하는 번호를 입력하세요.
1 : 전체 성적 출력
2 : 학생별 출력
3 : 성적 추가
4 : 성적 수정
5 : 성적 처리
0 : 종료
1
번호 이름 국어 수학 영어 총점 평균
  1  하나   78   90  100    0 0.00
  2  둘      8   98   67    0 0.00
  3  셋     78   90   88    0 0.00
  4  넷     78   88   90    0 0.00
  5  다섯   58   88   91    0 0.00
  6  여섯   77   70   76    0 0.00
  7  일곱  100   90    9    0 0.00
  8  여덟   98  100   87    0 0.00
  9  아홉   93   95   50    0 0.00
 10  열     66   90   88    0 0.00
```

프로그램 결과

```
원하는 번호를 입력하세요.
1 : 전체 성적 출력
2 : 학생별 출력
3 : 성적 추가
4 : 성적 수정
5 : 성적 처리
0 : 종료
5
성적 처리 완료

원하는 번호를 입력하세요.
1 : 전체 성적 출력
2 : 학생별 출력
3 : 성적 추가
4 : 성적 수정
5 : 성적 처리
0 : 종료
2
학생 id : 4
번호 :   4  이름 :    넷
  국어 :  78 수학 :  88 영어 :  90
  총점 : 256  평균 : 85.33
```

프로그램 결과

```
원하는 번호를 입력하세요.
1 : 전체 성적 출력
2 : 학생별 출력
3 : 성적 추가
4 : 성적 수정
5 : 성적 처리
0 : 종료
4
학생 id : 4
넷 학생 성적을 입력하세요 : 100 98 100

원하는 번호를 입력하세요.
1 : 전체 성적 출력
2 : 학생별 출력
3 : 성적 추가
4 : 성적 수정
5 : 성적 처리
0 : 종료
3
이름과 성적을 입력하세요 : 끝 100 100 100
```

프로그램 결과

```
원하는 번호를 입력하세요.
1 : 전체 성적 출력
2 : 학생별 출력
3 : 성적 추가
4 : 성적 수정
5 : 성적 처리
0 : 종료
1
번호 이름 국어 수학 영어 총점 평균
  1  하나   78   90  100  268 89.33
  2  둘      8   98   67  173 57.67
  3  셋     78   90   88  256 85.33
  4  넷    100   98  100  298 99.33
  5  다섯   58   88   91  237 79.00
  6  여섯   77   70   76  223 74.33
  7  일곱  100   90    9  199 66.33
  8  여덜   98  100   87  285 95.00
  9  아홉   93   95   50  238 79.33
 10  열     66   90   88  244 81.33
 11  끝    100  100  100  300 100.00
```