10장 입출력 함수

박 종 혁 교수

UCS Lab

Tel: 970-6702

Email: jhpark1@seoultech.ac.kr

printf()

- **특징**
  - 임의의 개수의 인자 출력
  - 간단한 변환 명세나 형식을 사용한 출력 제어

printf()

printf(*control_string, other_argument*)

- 예

  printf("she sells %d %s for $%f", 99,
  "sea shells", 3.77);

  control_string: "she sells %d %s for
  $%f"

  other_arguments: 99, "sea shells", 3.77

- 변환 명세는 %로 시작하여 변환문자로 끝남

printf()

printf() 변환 문자	
변환 문자	대응되는 인자의 출력 형태
c	문자
d,i	10진 정수
u	부호 없는 10진 정수
o	부호 없는 8진 정수
x, X	부호 없는 16진 정수
e	부동 소수점 수; 예: 7.123000e+00
E	부동 소수점 수; 예: 7.123000E+00
f	부동 소수점 수; 예: 7.123000
g	e형식과 f 형식 중 짧은 쪽
G	E형식과 f 형식 중 짧은 쪽
s	문자열
p	대응되는 인자가 void 포인터임; 그 값이 16진수 형태로 출력됨
n	대응되는 인자는 정수형 포인터로서 그 값은 현재까지 출력된 문자의 개수임; 인자는 변환되지 않음
%	%%의 형식으로 %를 출력 스트림에 씀; 대응되는 인자는 없음.

printf()

- **예제**

```
printf("she sells %d %s for $%f", 99,
  "sea shells", 3.77);
```

변환 형식	대응되는 인자
%d	99
%s	"sea shells"
%f	3.77

printf()

- **%와 변환문자 사이에 올 수 있는 것들**
  - 플래그 문자들
    - 빼기 기호
    - 더하기 기호
    - 공백
    - # 기호
    - 0
  - 필드 폭
  - 정밀도
  - h 또는 l
  - L

printf()

선언과 초기화			
char c = 'A', s[] = "Blue moon!";			
변환 형식	대응되는인자	필드 내에서 출력형태	비고
%c	c	"A"	필드 폭 1 (디폴트)
%2c	c	" A"	필드 폭 2, 우측 정렬
%-3c	c	"A "	필드 폭 3, 좌측 정렬
%s	s	"Blue moon!"	필드 폭 10 (디폴트)
%3s	s	"Blue moon!"	공간이 더 필요함
%.6s	s	"Blue m"	정밀도 6
%-11.8s	s	"Blue moo "	정밀도 8, 좌측 정렬

printf()

선언과 초기화			
int i = 123; double x = 0.123456789;			
변환 형식	대 응 되 는 인 자	필드 내에서 출력형태	비고
%d	i	"123"	필드 폭 3 (디폴트)
%05d	i	"00123"	영으로 채움
%7o	i	" 173"	우측 정렬, 8진수
%-9x	i	"7b "	좌측 정렬, 16진수
%-#9x	i	"0x7b "	좌측 정렬, 16진수
%10.5f	x	" 0.12346"	필드 폭 10, 정밀도 5
%-12.5e	x	"1.23457-01 "	좌측 정렬, e-형식

scanf()

scanf(*control_string, other_argument*)

- 예

```
char      a, b, c, s[100];
int       n;
double    x;
scanf("%c%c%c%d%s%lf",
  &a,&b,&c,&n,s,&x);
```

- *control_string*: "%c%c%c%d%s%lf"
- *other_arguments*: &a, &b, &n, s, &x

scanf()

scanf( ) 변환 문자		
변환 문자	입력 스트림에서 대응되는 문자	대응 인자의 포인터 형
c	공백을 포함한 모든 문자	char
d, i	10진 정수 (부호는 옵션)	integer
u	10진 정수 (부호는 옵션)	unsigned integer
o	8진수 (부호는 옵션)	unsigned integer
x, X	16진수 (부호는 옵션)	unsigned integer
e, E, f, g, G	실수 (부호는 옵션)	floating type
s	공백 없는 문자열	Char
p	printf() 함수의 %p에 의해 출력되는 것으로일반적으로 부호 없는 16진 정수임	void *
n, %, [...]	다음 표 참조	

scanf()

scanf( ) 변환 문자	
변환 문자	설명
n	입력 스트림의 문자와 짝을 이루지 않는다. 대응되는 인자는 정수형 포인터로서, 지금까지 읽어들인 문자의 개수를 저장한다.
%	입력 스트림에서 하나의 %와 짝을 이룬다. 대응되는 인자는 없다.
[...]	각괄호 [ ] 안에 있는 문자들을 스캔 집합이라 한다. 이것은 무엇이 짝을 이루는가를 결정한다. (아래 설명을 참조하여라.) 대응되는 인자는 문자 배열의 기본 주소에 대한 포인터이고, 이 배열은 끝에 자동적으로 추가되는 널 문자를 포함하여 대응되는 모든 문자들을 포함할 만큼 큰 크기를 가져야 한다.

scanf()

- **제어 문자열은 다음과 같은 것을 포함할 수 있음**
  - 여백
  - %이외의 공백문자가 아닌 일반 문자
  - %로 시작해서 변환 문자로 끝나는 변환 명세
  - h
  - l
  - L

scanf() 예제

```
int    i;
char   c;
char   string[15];
scanf("%d , %*s %% %c %5s %s", &i, &c,
  string, &string[5]);
```

* 입력 스트림 : 45 , ignore_this % C read_in_this**
  - i : 45
  - c : C
  - string[0-5] : "read_"
  - string[5-14] : "in_this**"
  - scanf()는 4를 리턴

sprintf()/sscanf()

- **각각 printf()와 scanf() 함수의 문자열 버전**
- **함수 원형**

```
int sprintf(char *, const char *, ...);
int sscanf(const char *, const char *, ...);
```

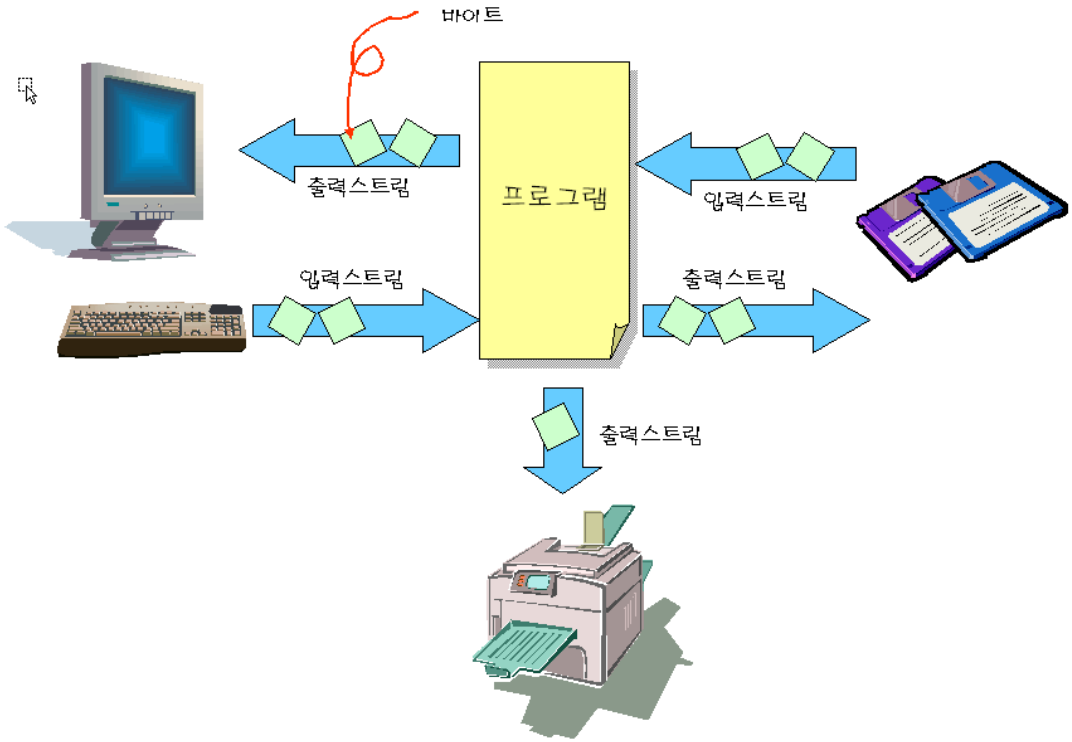
파일 입출력의 개념

- 파일은 데이터를 입출력하는 모든 대상을 의미
  - 키보드로부터 데이터를 입력하고 모니터로 출력하는 것은 키보드 파일과 모니터파일로 데이터를 입출력하는 것임

  - 프로그램은 사실상 **스트림 파일(stream file)**이라고 하는 표준화된 형태의 파일로 입출력을 수행하고 이 파일이 다시 물리적인 장치와 연결되어 실제적인 입출력이 수행됨

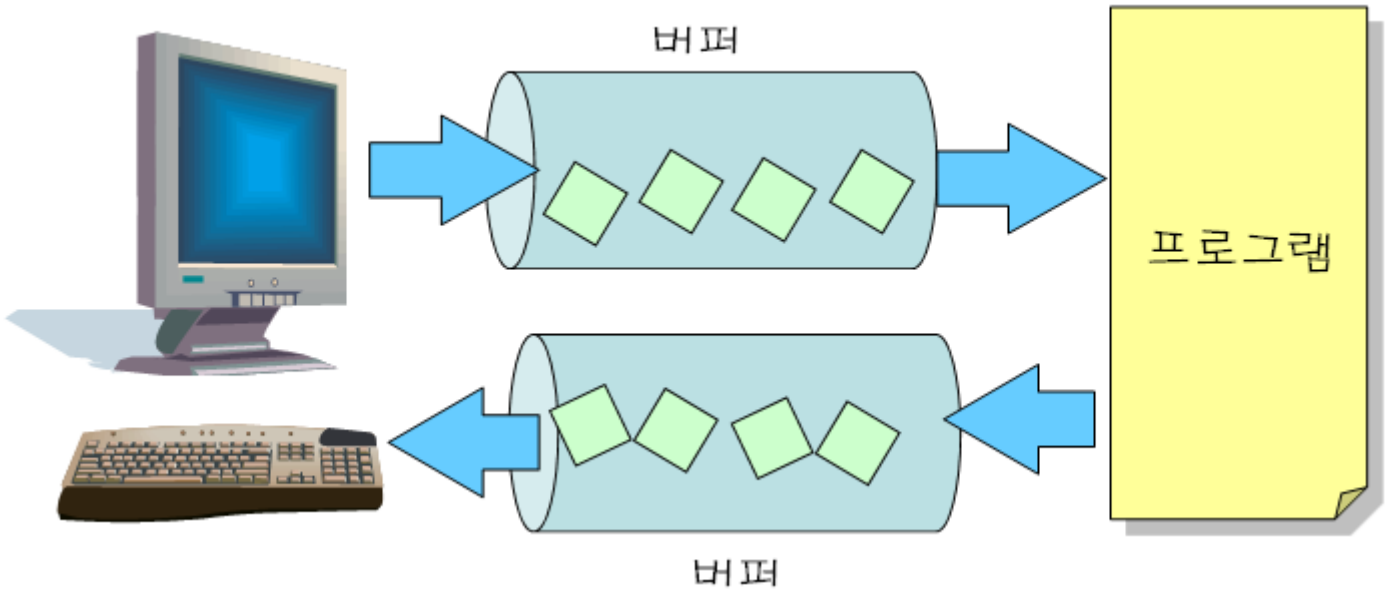
스트림 파일을 사용하는 이유

- 입출력 함수들이 다양한 입출력장치와 독립적으로 일관된 입출력 작업을 해야 한다(입출력 장치는 항상 변한다).
- 입력과 출력을 바이트(byte)들의 흐름으로 생각하는 것



스트림 파일을 사용하는 이유

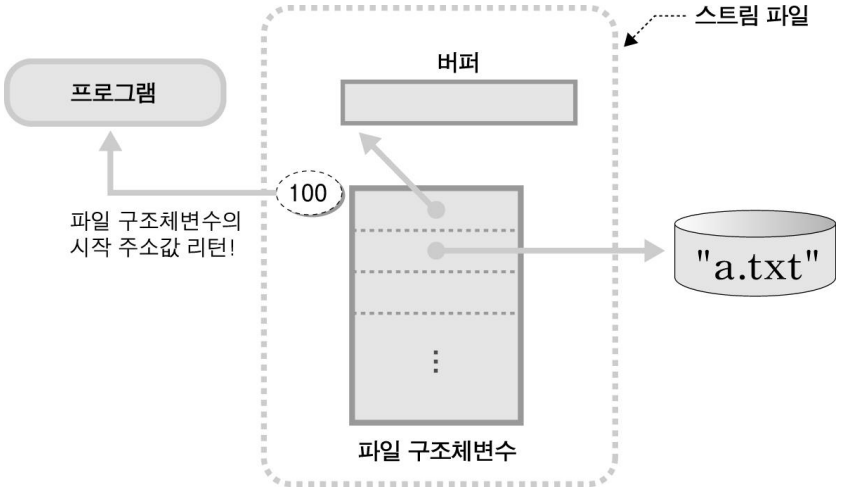
- 프로그램에서 데이터를 처리하는 속도와 입출력 장치에서 수행되는 입출력 속도의 차이를 줄이는 역할을 한다.
  - 하드디스크의 처리속도는 메모리의 전기적 처리속도를 따라갈 수 없다.
  - 스트림파일은 버퍼(buffer)를 사용하여 속도차이를 줄인다.



fopen()

- fopen함수가 개방에 성공하면 스트림파일을 만들고 파일포인터를 리턴한다.

- **스트림파일은 데이터를 저장하는 버퍼와 버퍼를 관리하는 여러 정보를 파일 구조체변수에 저장하고 있는데 이 구조체변수의 포인터가 파일포이터이다.**



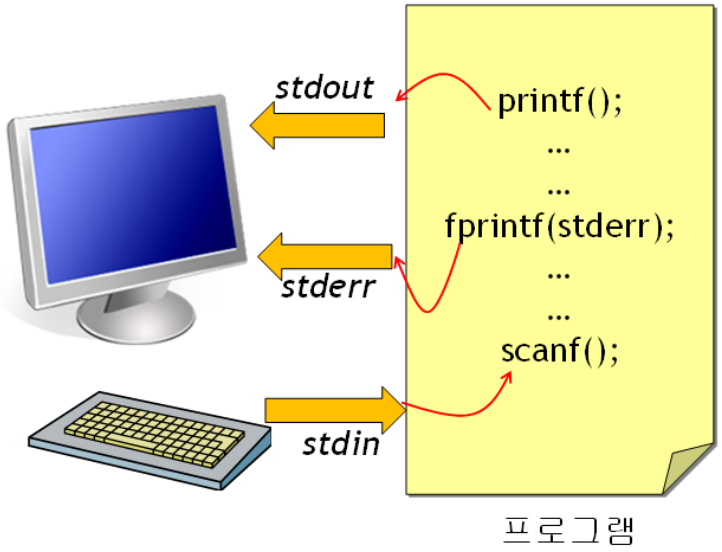

```
struct _iobuf{
   char  *_ptr;
   int   _cnt;
   char  *_base;
   int   _flag;
   int   _file;
   int   _charbuf;
   int   _bufsize;
   char  *tmpfname;
};
```

typedef struct _iobuf FILE;

표준 입출력 스트림

- 기본적인 스트림들은 프로그래머가 생성하지 않아도 자동으로 생성된다.

이름	*스트림*	*연결 장치*
stdin	표준 입력 스트림	키보드
stdout	표준 출력 스트림	모니터의 화면
stderr	표준 오류 스트림	모니터의 화면



fopen()

- 파일포인터를 포인터변수에 저장하면 입출력 준비작업이 끝난다.

```
FILE *fp;                          // FILE구조체를 가리키는 포인터변수
fp = fopen("a.txt", "w");  // 파일포인터를 포인터변수에 저장한다.
```

- **파일 개방에 실패하면 fopen함수는 널 포인터를 리턴한다.**

- 널 포인터를 사용하면 실행할 때 에러가 발생하므로 반드시 개방에 성공했는지를 검사해야 한다.

```
fp = fopen("b.txt", "r");          // 읽기 전용으로 파일 개방
if(fp == NULL){                    // 파일이 개방되지 않았으면 조건식은 참
   printf("파일이 없습니다.");    // 안내 메시지를 출력하고
   return 1;                       // 프로그램을 종료한다.
}
```

fopen()

- 스트림 파일을 만드는 것을 파일 개방이라고 하며 fopen함수를 사용하여 수행한다.

```
FILE *fopen(char *, char *);      // file open, 파일 개방
```

- **출력 전용으로 사용할 파일을 개방하는 예**

```
fopen( "a.txt", "w" );   // 파일 개방 함수 호출
```

개방할 파일 이름 ····· "a.txt"

"w" ····· 출력 전용(write)으로 개방

- **개방할 파일은 현재의 작업 디렉토리에서 찾으며 경로를 직접 지정할 수도 있다.**

```
fopen("c:\\source\\a.txt", "w");
```

// 디렉토리를 표시하는 백슬래시는 문자열 안에 있으므로 두 번 사용한다.

fopen()

- fopen(filename, mode) 형태의 함수 호출은 filename 파일을 mode에 지정된 모드로 열고, 파일 포인터를 리턴함

- **모드**

모드	설명
“r”	읽기 모드로 파일을 연다.
“w”	쓰기 모드로 파일을 생성한다. 만약 파일이 존재하지 않으면 파일이 생성된다. 파일이 이미 존재하면 기존의 내용이 지워진다.
“a“	추가 모드로 파일을 연다. 만약 똑같은 이름의 기존의 파일이 있으면 데이터가 파일의 끝에 추가된다. 파일이 없으면 새로운 파일을 만든다.
“r+”	읽기 모드로 파일을 연다. 쓰기 모드로 전환할 수 있다. 파일이 반드시 존재하여야 한다.
“w+”	쓰기 모드로 파일을 생성한다. 읽기 모드로 전환할 수 있다. 파일이 존재하면 기존의 데이터가 지워진다.
“a+”	추가 모드로 파일을 연다. 읽기 모드로 전환할 수 있다. 데이터를 추가하면 EOF 마커를 추가된 데이터의 뒤로 이동한다. 파일이 없으면 새로운 파일을 만든다.
“b”	이진 파일 모드로 파일을 연다.

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

fopen()

- 출력전용 모드는 같은 이름의 파일이 있을 때 그 내용을 모두 삭제하고 개방하므로 주의해야 한다.

  - 일단 읽기전용 모드로 개방한 후에 파일 존재여부를 확인하고 다시 출력전용으로 개방한다.

```
ifp=fopen("a.txt", "r");          // 일단 읽기 전용으로 개방한다.
if(ifp==NULL){                    // 파일이 없으면 조건식은 참
   ofp=fopen("a.txt", "w");       // 이 때 다시 쓰기 전용으로 개방한다.
}
```

fopen()/fclose()

- **파일 열기와 닫기의 전형적인 예제 코드**

```
#include <stdio.h>
int main(void){
    int     a, sum = 0;
    FILE    *ifp, *ofp;
    ifp = fopen("my_file", "r");
    ofp = fopen("outfile", "w");
    .....
    fclose(ifp);
    fclose(ofp);
}
```

fopen()/fclose()

```
#include <stdio.h>                  // FILE구조체에 대한 형 선언이 포함되어 있다.
int main()
{
   FILE *ifp, *ofp;                 // FILE구조체 포인터변수 선언
   ifp=fopen("a.txt", "r");         // a.txt 파일을 읽기 전용으로 개방
   if(ifp==NULL){                   // 파일이 없으면 조건식은 참
      printf("입력파일이 개방되지 않았습니다.\n");     // 안내 메시지 출력
      return 1;                                      // 프로그램 종료
   }
   printf("입력파일이 개방되었습니다.\n");
   ofp=fopen("b.txt", "w");         // b.txt 파일은 쓰기 전용으로 개방한다.
   if(ofp==NULL){
      printf("출력파일이 개방되지 않았습니다.\n");
      return 1;
   }
   printf("출력파일이 개방되었습니다.\n");
   return 0;
}
```

fopen()/fclose()

- 사용이 끝난 파일은 파일을 닫아서 스트림파일을 제거한다.

```
int fclose(FILE *);     // file close, 파일 종결
```

- 성공적으로 닫으면 0을 리턴하며 오류가 발생하면 -1을 리턴한다.

```
FILE *fp;
int res;                        // fclose함수의 리턴값을 저장할 변수
fp=fopen("a.txt", "r");  // 파일 개방
…
res=fclose(fp);                 // 파일포인터변수 fp를 전달인자로 주고 파일을 닫는다.
if(res!=0){
   printf("파일이 닫히지 않았습니다.\n");
   return 1;
}
```

- 개방된 파일은 프로그램이 종료되면 자동으로 닫히면서 메모리에서 제거 되지만 안정성을 위해서 명시적으로 닫는 것이 좋다.

fprintf()/fscanf()

- 각각 printf()와 scanf() 함수의 파일 버전
- 함수 원형

```
int fprintf(FILE *fp, const char *format, …)
int fscanf(FILE *fp, const char *format, …)
```

- fprintf(stdout,…);와 printf(…);는 같은 의미
- fscanf(stdin, …);은 scanf(…);와 같은 의미

fprintf()/fscanf()

- **fscanf, fprintf함수는 scanf, printf함수와 사용법이 같다. 단, 입출력 대상을 파일포인터로 지정해 줄 수 있다.**

```
int fscanf(FILE *, char *, …);   // 파일에서 형식에 따라 데이터 입력
int fprintf(FILE *, char *, …);  // 파일로 형식에 따라 데이터 출력
```

- **fscanf함수는 데이터의 입력이 끝나면 -1을 리턴한다.**

- **예제) 이름, 나이, 키가 저장된 텍스트 파일의 데이터를 형식에 따라 입력한 후에 키, 나이, 이름의 순서로 출력하는 예**

텍스트 파일 a.txt

```
박준성 25 188.9
지혜연 23 162.5
조충근 19 175.0
```

텍스트 파일 b.txt

```
188.9 25 박준성
162.5 23 지혜연
175.0 19 조충근
```

fprintf()/fscanf()

```
#include <stdio.h>

int main()
{
   FILE *ifp, *ofp;
   char name[20];
   int age;
   double height;
   int res;

   ifp=fopen("a.txt", "r");
   if(ifp==NULL){
      printf("입력파일 개방 실패.\n");
      return 1;
   }
   ofp=fopen("b.txt", "w");
   if(ofp==NULL){
      printf("출력파일 개방 실패.\n");
      return 1;
   }
   while(1){
      res=fscanf(ifp, "%s%d%lf", name, &age, &height);
      if(res==EOF) break;
      fprintf(ofp, "%.1lf %d %s\n", height, age, name);
   }

   fclose(ifp);
   fclose(ofp);
   return 0;
}
```

fputc()

- **문자 하나를 파일에 출력할 때는 fputc함수를 사용한다.**

```
int fputc(int, FILE *);     // 하나의 문자를 파일로 출력한다.
```

- 첫 번째 전달인자로 주어지는 문자를 두 번째 전달인자의 파일로 출력한다.

- **키보드로부터 입력되는 데이터를 파일로 출력하는 예**

```
#include <stdio.h>

int main()
{
   FILE *fp;
   char ch;
   fp=fopen("b.txt", "w");
   if(fp==NULL){
      printf("파일 개방 실패.\n");
      return 1;
   }
   printf("데이터를 입력하세요.\n");
   while(1){
      ch=getchar();          // 키보드 입력
      if(ch==EOF) break;
      fputc(ch, fp);         // 파일로 출력
   }
   fclose(fp);
   return 0;
}
```

```
데이터를 입력하세요.
banana (엔터)
apple (엔터)
^Z (입력 종료)
```

출력결과는 b.txt 파일을 메모장으로 열어 확인해 보도록 합시다.

하드디스크 파일 b.txt

```
banana
apple
```

fgets()

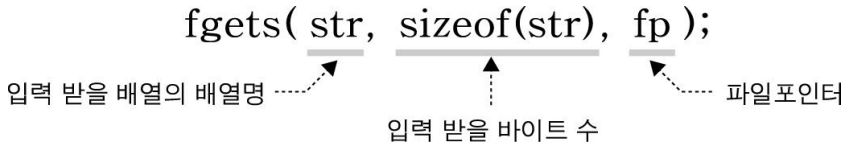
- **문자열을 한번에 입력 할 때는 fgets함수를 사용한다.**

```
char *fgets(char *, int, FILE *);   // 파일에서 문자열을 읽어 들인다.
```

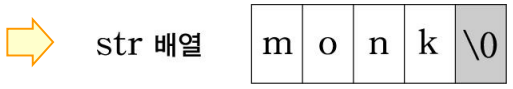
- 파일포인터와 연결된 파일로부터 두 번째 전달인자로 주어진 바이트 수에 따라 데이터를 읽어와서 첫 번째 전달인자로 주어진 배열에 저장한다.

- 5바이트의 크기를 갖는 배열에 문자열을 입력 받는 경우

```
FILE *fp;
char str[5];
fp=fopen("a.txt", "r");
```



널문자가 저장될 공간을 제외하고 4바이트만 입력된다.



fgets()

- **fgets함수는 문자열 입력에 앞서 줄 단위로 입력 받는다.**
  - **배열의 크기가 10바이트인 경우의 입력(새줄 문자도 입력 받는다).**

```
FILE *fp;
char str[10];
fp=fopen("a.txt", "r");
fgets(str, sizeof(str), fp);
```

str 배열

m	o	n	k	e	y	\n	\0	?	?

  - **입력 받은 문자열에서 새줄 문자가 불필요할 때에는 제거한다.**

```
str[strlen(str)-1] = '\0';
```

str 배열에서 새줄 문자가 저장된 위치의 첨자가 구해진다.

strlen(str) ➡ 널문자 전까지의 바이트 수 ➡ 7

strlen(str) -1 ➡ 6 ➡ 새줄 문자가 저장된 위치의 첨자

str[strlen(str) -1] =\'0'; ➡ 새줄 문자가 널문자로 바뀐다.

fgets()

- 입력 받을 데이터의 수보다 파일의 크기가 작으면 파일 끝까지 읽어 들인다(물론 중간에 새줄 문자는 없어야 한다).

```
char str[80];
…
fgets(str, sizeof(str), fp);
```

입력파일의 데이터

```
monkey and tiger
```

⇨ 입력파일의 데이터가 80바이트가 안되므로 파일 끝까지 입력된다!

- fgets함수의 리턴값을 입력한 배열의 포인터이다. 따라서 입력이 끝난 후에 바로 이 포인터를 사용하여 문자열을 출력할 수 있다.

```
printf("%s\n", fgets(str, sizeof(str), fp));
```

- 입력파일에서 더 이상 읽어 들일 데이터가 없으면 널 포인터를 리턴한다. (-1(EOF)가 아니므로 주의한다!)

```
…
res=fgets(str, sizeof(str), fp);
if(res==NULL) break; // 파일의 끝이면 입력을 종료한다.
…
```

fputs()

- 문자열을 한번에 출력 할 때는 **fputs**함수를 사용한다.

```
int fputs(char *, FILE *);     // 파일로 문자열을 출력한다.
```

- 첫 번째 전달인자는 출력할 문자열의 위치를 주고 두 번째 전달인자는 파일 포인터를 준다(puts함수와는 달리 자동으로 줄을 바꾸지 않는다).

```
FILE *fp;                      // 파일포인터변수
char str[] = "orange";         // 출력할 데이터가 저장된 배열, 초기화한다.
fp=fopen("b.txt", "w");        // 파일을 출력용으로 개방
```

```
fputs ( str,  fp );
```

출력할 배열의 배열명 ↑ ↑ 파일포인터

b.txt파일

orange

예제

```
#include <stdio.h>
#include <string.h>

int main()
{
   FILE *ifp, *ofp;
   char str[80];
   char *res;

   ifp=fopen("a.txt", "r");
   if(ifp==NULL){
      printf("입력파일 개방 실패.\n");
      return 1;
   }

   ofp=fopen("b.txt", "w");
   if(ofp==NULL){
      printf("출력파일 개방 실패.\n");
      return 1;
   }

   while(1){
      res=fgets(str, sizeof(str), ifp);
      if(res==NULL) break;
      str[strlen(str)-1]='\0';
      fputs(str, ofp);
      fputs(" ", ofp);
   }

   fclose(ifp);
   fclose(ofp);

   return 0;
}
```

입력 파일 a.txt

```
소년은 (엔터)
늙기 쉽고 (엔터)
학문은 (엔터)
이루기 어렵다. (엔터)
```

출력 파일 b.txt

```
소년은 늙기 쉽고 학문은 이루기 어렵다.
```

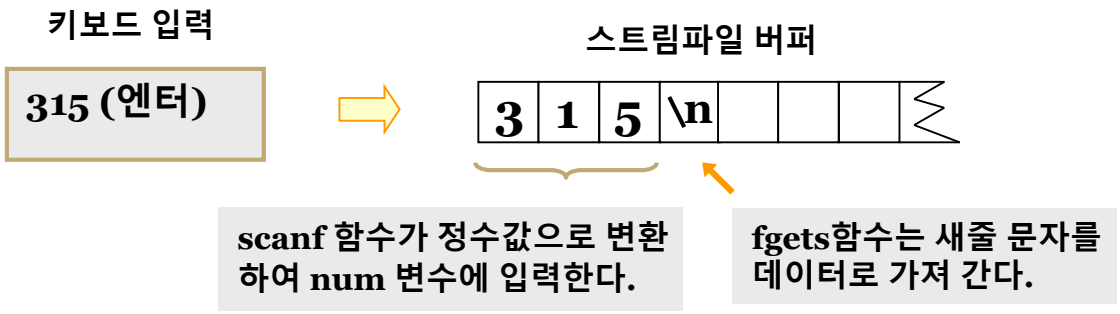
fflush()

- 입출력 함수들이 버퍼를 공유하기 때문에 예상치 못한 문제가 발생한다.
  - 학번을 입력하고 바로 이름을 입력하는 예

```
int num;
char name[20];
printf("학번을 입력하세요 : ");
scanf("%d", &num);
printf("이름을 입력하세요 : ");
fgets(name, sizeof(name), stdin);
```

```
학번을 입력하세요 : 315 (엔터)
이름을 입력하세요 : 학번 : 315
이름 :
```

- scanf함수가 학번을 입력 받은 후에 버퍼에 남겨진 새줄 문자를 다음에 호출되는 fgets함수가 데이터로 받아들이기 때문이다.



fflush()

- **fflush함수는 버퍼에 남아 있는 불필요한 데이터를 삭제한다.**

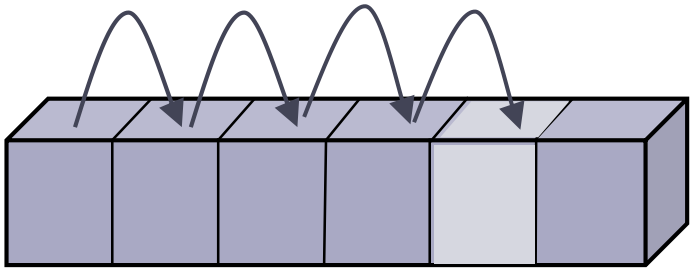
```
int fflush(FILE *);      // 스트림파일의 버퍼를 비워준다.
```

```
int num;
char name[20];
printf("학번을 입력하세요 : ");
scanf("%d", &num);
fflush(stdin);    // scanf함수와 gets함수가 공유하는 표준 입력 스트림버퍼를 비운다.
printf("이름을 입력하세요 : ");
fgets(name, sizeof(name), stdin);
```

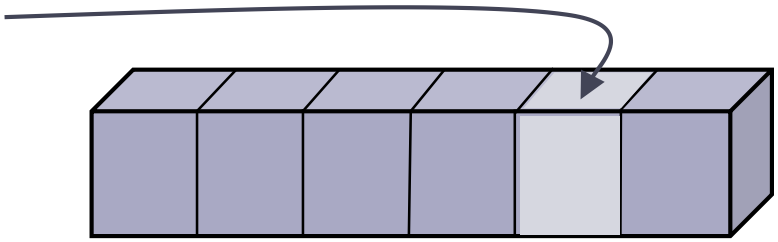
```
학번을 입력하세요 : 315 (엔터)
이름을 입력하세요 : 홍 길동 (엔터)
학번 : 315
이름 : 홍 길동
```

임의 접근 파일

- **순차 접근(sequential access)** 방법: 데이터를 파일의 처음부터 순차적으로 읽거나 기록하는 방법
- **임의 접근(random access)** 방법: 파일의 어느 위치에서든지 읽기와 쓰기가 가능한 방법



순차접근파일



임의접근파일

파일의 임의의 위치 접근

- `ftell(file_ptr)`
  - 파일 위치 지시자의 현재 값을 리턴
- `fseek(file_ptr, offset, place)`
  - 파일 위치 지시자를 place부터 offset 바이트 떨어진 곳을 나타내는 값으로 설정함
  - place의 값은 0(SEEK_SET), 1(SEEK_CUR), 2(SEEK_END) 중 하나가 될 수 있는데, 이것들은 각각파일의 처음, 현재 위치, 파일의 끝을 나타냄

파일의 임의의 위치 접근

- **파일을 역으로 출력하는 프로그램**

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

정리

파일의 정의와 종류

파일(File)
: 프로그램에서 사용할 데이터를 보조 기억장치에 저장한 데이터들의 집합
(예) 문서, 소리, 그림, 동영상 등과 같은 모든 종류의 자료

- **텍스트 파일**
  ① 한글을 제외한 모든 문자를 ASCII 코드로 표현
  ② 일반적으로 문서 파일이며, 아무 연산 없이 읽을 수 있는 문자가 들어 있는 파일
  ③ 각 라인은 라인의 끝을 표시하는 문자로 종료
  ④ 순차적으로 입출력할 수 있지만 랜덤 입출력 불가

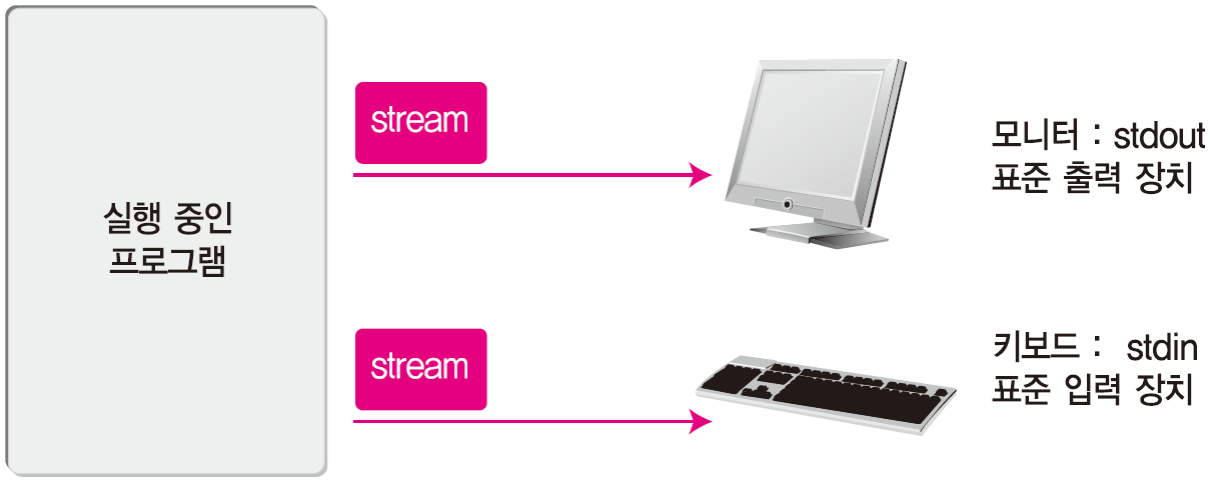
- **이진 파일**
  ① ASCII 코드가 아닌 2진수의 형태로 저장하는 모든 파일
  ② 문자가 아닌 이진 데이터로 저장된 파일
  ③ 모든 데이터들은 변환없이 입출력되고, 라인을 분리하지 않음
  ④ 실행 파일, 사운드 파일, 그래픽 파일 등은 이진 파일
  ⑤ 랜덤하게 읽거나 쓰기 가능

버퍼

버퍼(buffer)

- 데이터를 일정량 저장할 수 있는 메모리 공간
- 키보드를 통해 데이터를 입력하면 [Enter] 키를 누를 때까지 스트림을 통해 입력 버퍼에 보관하다가 [Enter] 키를 누르면 입력 버퍼의 내용들을 실행 중인 프로그램으로 전달

스트림



파일의 끝 EOF

- C언어는 파일을 열어 더 이상 읽을 데이터가 없으면 '텍스트 파일의 끝'을 의미하는EOF(End Of File)을 반환
- EOF는 stdio.h 헤더 파일에 다음과 같이 정의

```
#define EOF (-1)
```

- DOS에서는 [Ctrl]+[Z] 키를 누르면 EOF 문자로 인식
- UNIX나 LINUX에서는 [Ctrl]+[C] 키를 누르면 EOF 문자로 인식
- 이진 파일일 경우에는 파일의 끝이라는 의미로 사용되지 않고 일반 데이터로 처리됨
- 텍스트 파일과 이진 파일에서는 feof( ) 함수를 사용하여 파일의 끝(파일의 끝에 도달하면 0이 아닌 값을 리턴) 확인 가능

파일 입출력

파일에 데이터 입출력을 위한 작성 순서

① 파일명 대신 파일 포인터를 사용

FILE *파일 포인터;

② 파일을 읽기와 쓰기 중 어떤 용도로 사용할 것인지를 결정하여 "모드"에 기술하고 파일을 열기

파일 포인터 = fopen("파일명", "모드");

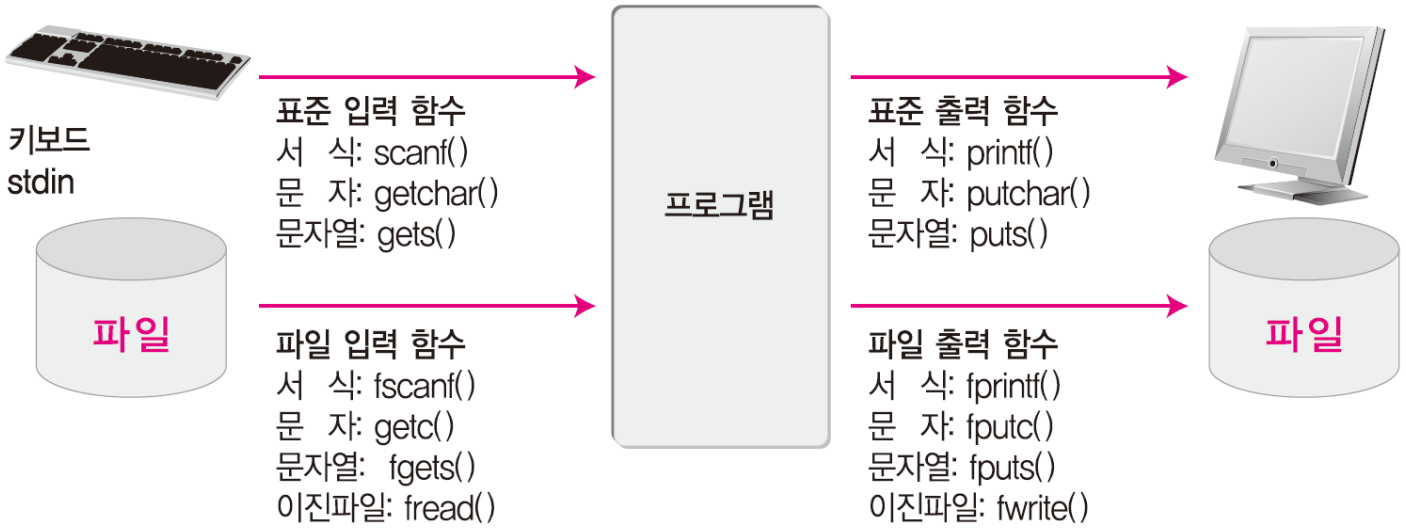
③ 파일 처리: 파일 입출력

④ 파일 닫기: fclose(파일 포인터 변수);

FILE 구조체는 stdio.h 헤더파일에 선언

스트림에 접근하기 위한 자료구조로 각 스트림 마다 자신만의 FILE 구조체를 갖음

텍스트 파일 입출력 함수



사용 빈도수 높은 파일 입출력 함수

- **fgetc( ), getc( ), getchar( ) 함수**
  : unsigned char 타입으로 읽은 문자를 int 타입으로 반환, 파일의 끝이면 EOF, 에러가 발생하면 에러 값을 반환

- **gets( )와 fgets( ) 함수**
  : 성공하면 문자열의 포인터를 반환, 파일의 끝이거나 에러가 발생한 경우 또는 아무런 문자도 입력받지 못하면 EOF를 반환

연습예제

예제1

```c
#include <stdio.h>
#include <string.h>

int main(void)
{
    FILE *fp;
    char fname[128];
    char buffer[256];
    char word[256];
    int line_num = 0;

    printf("입력 파일 이름을 입력하시오: ");
    scanf("%s", fname);

    printf("탐색할 단어를 입력하시오: ");
    scanf("%s", word);
```

proverb.txt

```
A chain is only as strong as its weakest link
A change is as good as a rest
A fool and his money are soon parted
A friend in need is a friend indeed
A good beginning makes a good ending
A little knowledge is a dangerous thing
...
```

```
    // 파일을 읽기 모드로 연다.
    if( (fp = fopen(fname, "r")) == NULL )
    {
            fprintf(stderr,"파일 %s을 열 수 없습니다.\n", fname);
            exit(1);
    }
    while( fgets(buffer, 256, fp) )
    {
            line_num++;
            if( strstr(buffer, word) )
            {
                printf("%s: %d 단어 %s이 발견되었습니다.\n", fname, line_num, word );
            }
    }
    fclose(fp);
    return 0;
}
```

```
입력 파일 이름을 입력하시오: proverb.txt
탐색할 단어를 입력하시오: house
proverb.txt: 7 단어 house이 발견되었습니다.
proverb.txt: 8 단어 house이 발견되었습니다.
```

예제2. 성적 파일 입출력

```
int main(void)
{
    FILE *fp;
    char fname[100];
    int number, count = 0;
    char name[20];
    float score, total = 0.0;

    printf("성적 파일 이름을 입력하시오:  ");
    scanf("%s", fname);

    // 성적 파일을 쓰기 모드로 연다.
    if( (fp = fopen(fname, "w")) == NULL )
    {
        fprintf(stderr,"성적 파일 %s을 열 수 없습니다.\n", fname);
        exit(1);
    }
```

```
// 사용자로부터 학번, 이름, 성적을 입력받아서 파일에 저장한다.
while( 1 )
{
        printf("학번, 이름, 성적을 입력하시요: (음수이면 종료)");
        scanf("%d", &number);
        if( number < 0 ) break
        scanf("%s %f", name, &score);
        fprintf(fp, "%d %s %f\n", number, name, score);
}
fclose(fp);
// 성적 파일을 읽기 모드로 연다.
if( (fp = fopen(fname, "r")) == NULL )
{
        fprintf(stderr,"성적 파일 %s을 열 수 없습니다.\n", fname);
        exit(1);
}
```

```
        // 파일에서 성적을 읽어서 평균을 구한다.
    while( !feof( fp ) )
    {
        fscanf(fp, "%d %s %f", &number, name, &score);
        total += score;
        count++;
    }
    printf("평균 = %f\n", total/count);
    fclose(fp);
    return 0;
}
```

```
성적 파일 이름을 입력하시오:  score.txt
학번, 이름, 성적을 입력하시요: (음수이면 종료) 1 KIM 90.2
학번, 이름, 성적을 입력하시요: (음수이면 종료) 2 PARK 30.5
학번, 이름, 성적을 입력하시요: (음수이면 종료) 3 MIN 56.8
학번, 이름, 성적을 입력하시요: (음수이면 종료)-1
평균 = 58.575001
```

예제3. 임의 접근 파일

```
#include <stdio.h>
#include <stdlib.h>

#define SIZE 1000
void init_table(int table[], int size);

int main(void)
{
	int table[SIZE];
	int n, data;
	long pos;
	FILE *fp = NULL;

	// 배열을 초기화한다.
	init_table(table, SIZE);

	// 이진 파일을 쓰기 모드로 연다.
	if( (fp = fopen("sample.dat", "wb")) == NULL )
	{
		fprintf(stderr,"출력을 위한 파일을 열 수 없습니다.\n");
		exit(1);
	}
```

```
    // 배열을 이진 모드로 파일에 저장한다.
    fwrite(table, sizeof(int), SIZE, fp);
    fclose(fp);
// 이진 파일을 읽기 모드로 연다.
    if( (fp = fopen("sample.dat", "rb")) == NULL )
    {
        fprintf(stderr,"입력을 위한 파일을 열 수 없습니다.\n");
        exit(1);
    }
    // 사용자가 선택한 위치의 정수를 파일로부터 읽는다.
    while(1)
    {
        printf("파일에서의 위치를 입력하십시요(0에서 %d, 종료-1): ", SIZE - 1);
        scanf("%d", &n);
        if( n == -1 ) break
        pos = (long) n * sizeof(int);
        fseek(fp, pos, SEEK_SET);
        fread(&data, sizeof(int), 1, fp);
        printf("%d 위치의 값은 %d입니다.\n", n, data);
    }
    fclose(fp);
    return 0;
}
```

```
// 배열을 인덱스의 제곱으로 채운다.
void init_table(int table[], int size)
{
      int i;

      for(i = 0; i < size; i++)
            table[i] = i * i;
}
```

```
파일에서의 위치를 입력하십시요(0에서 999, 종료 -1): 3
3 위치의 값은 9입니다.
파일에서의 위치를 입력하십시요(0에서 999, 종료 -1): 9
9 위치의 값은 81입니다.
파일에서의 위치를 입력하십시요(0에서 999, 종료 -1): -1
```

연습예제-Advanced

Adv. 1

사용자로부터 직원에 대한 정보를 받아서 employee.txt 파일에 저장하는 프로그램을 작성 하시오.

Prgoram

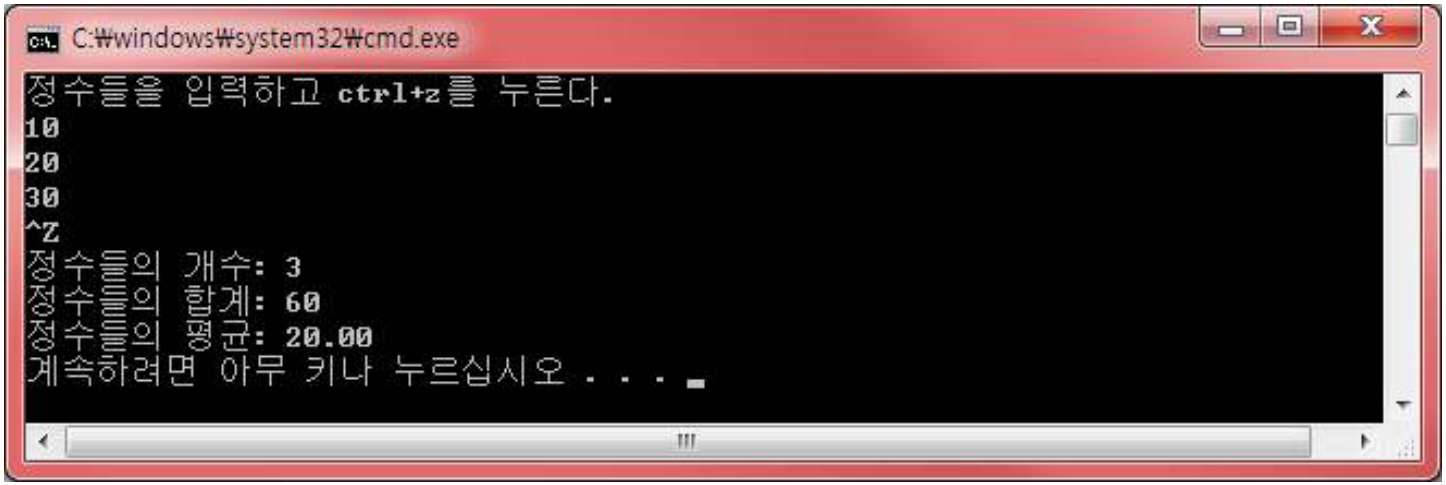
```
#include <stdio.h>
int main(void)
{
        FILE *fptr;
        char name[20];
        int age;
        float salary;
        fptr = fopen ("employee.txt", "w");

        if (fptr == NULL)
        {
                printf("파일을 생성할 수 없음!\n");
                return;
        }

        printf("직원 이름: ");
        scanf("%s", name);
        fprintf(fptr, "이름 = %s\n", name);
        printf("나이: ");
        scanf("%d", &age);
        fprintf(fptr, "나이 = %d\n", age);
        printf("월급: ");
        scanf("%f", &salary);
        fprintf(fptr, "월급 = %f\n", salary);
        fclose(fptr);
        return 0;
}
```

Adv. 2

정수들이 저장된 파일에서 모든 정수를 읽어서 정수의 개수, 합계, 평균을 출력하는 프로 그램을 작성하시오.



Prgoram 2-1

```
#include <stdio.h>
int main (void)
{
        int number, sum, count;
        double average;
        FILE *in, *f;
        char ch;
        f=fopen("numbers","w");
        printf("정수들을 입력하고 ctrl+z를 누른다.\n");

        do
        {
                ch=getchar();
                putc(ch,f);
        }

        while(ch!=EOF);
        fclose(f);
        in = fopen ("numbers", "r");
        sum = 0;
        count = 0;
```

Prgoram2-2

```
        while (fscanf (in, "%d", &number) == 1)
        {
                sum = sum + number;
                count = count + 1;
        }

        fclose (in);
        printf ("정수들의 개수: %d\n", count);
        printf ("정수들의 합계: %d\n", sum);
        average = (double)sum / count;
        printf ("정수들의 평균: %5.2lf\n", average);
        return (0);
}
```

참고문헌

- 열혈 C 프로그래밍, 윤성우, 오렌지미디어
- 쉽게 풀어쓴 C언어 Express, 천인국, 생능출판사
- 뇌를 자극하는 C 프로그래밍, 서현우, 한빛미디어
- 쾌도난마 C프로그래밍,강성수,북스홀릭
- C프로그래밍 기초와 응용실습, 고응남, 정익사

질의 및 응답