

고급C프로그래밍

07 표준/파일 입출력(1/2)

공지 사항

• 중간고사

- ✓ 일시 : 10/28(토) 11:00 ~ 12:00
- ✓ 장소 : 공지 예정(강의자료 게시판 참조)
- ✓ 시험범위 : 포인터/배열/함수, 고급 포인터, 표준/파일 입출력

표준 입출력 함수

- 개요

- **표준 입력(stdin, Standard Input) : 키보드 입력**
- **표준 출력(stdout, StandardOutput) : 모니터 출력**

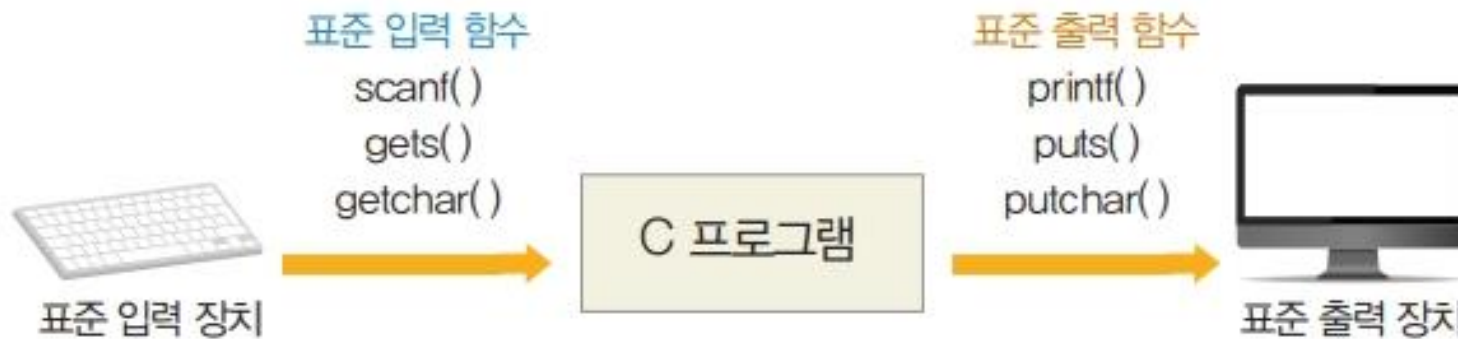


그림 11-1 표준 입출력의 개념

표준 입출력 함수

- 서식화된 입출력 함수

- **printf(), scanf()**

표 11-1 서식화된 입출력 함수 종류

구문	설명
printf("서식", 출력할 매개변수들 ...)	표준 출력 장치(모니터)에 서식을 맞춰 출력한다.
scanf("서식", 입력할 매개변수들 ...)	표준 입력 장치(키보드)에서 서식에 맞춰 입력받는다.

- **서식의 위치에 올 수 있는 것들**

표 11-2 입출력 함수에 사용 가능한 서식

서식	설명
%d	정수형(int)
%c	문자형(char)
%s	문자열(char*) 또는 문자 배열(char[])
%x	16진수 정수(int)
%o	8진수 정수(int)
%f, %lf	실수형(float, double)
%e	공학 계산용 형식
%p	포인터 주소

표준 입출력 함수

- 문자열 입출력 함수 (1/3)

- **printf(), scanf() 함수 : 모든 데이터 형식의 입출력**
- **puts(), gets() 함수 : 문자열의 입출력**

표 11-3 문자열 입출력 함수 종류

함수	설명
gets(문자 배열)	표준 입력 장치(키보드)로부터 '문자 배열 크기 - 1'만큼 문자열을 입력받는다. 숫자를 입력해도 무조건 문자열로 취급한다.
puts(문자 배열)	표준 출력 장치(모니터)에 문자열을 출력한다.

- **문자열만 입출력할 경우에는 printf(), scanf() 함수보다는 gets()(또는 gets_s()), puts() 함수를 사용하는 것이 처리 속도가 더 빠름**

표준 입출력 함수

- 문자열 입출력 함수 (2/3)

- **gets_s()**

- ✓ scanf()는 공백이 나타나면 문자열 입력을 종료
 - ✓ gets_s()는 '\n'이 나올 때까지(즉, 사용자가 Enter를 입력할 때까지) 입력받음
 - ✓ 최대 입력 문자는 널 문자를 고려해서 '배열크기 -1'까지 입력
- ➔ 마지막의 '\n'은 널 문자로 대체됨

```
char ss[10];  
gets_s(ss, sizeof(ss));
```

표준 입출력 함수

- 문자열 입출력 함수 (3/3)

- **puts()**

✓ '\n'이 없어도 출력한 후 자동으로 줄을 넘김

```
char ss[] = "XYZ"  
puts(ss);
```


표준 입출력 함수

- 문자 입출력 함수 (1/9)

- **getchar(), getch(), getche() 함수 : 문자 하나만 입력하는 기능**
- **putchar(), putch() 함수 : 문자 하나를 출력하는 기능**

표 11-4 문자 입출력 함수 종류

입력 함수	설명
getch()	키보드를 통해 문자 하나를 입력받으며, 입력한 내용을 모니터에 보여주지 않는다.
getche()	키보드를 통해 문자 하나를 입력받으며, 입력한 내용을 모니터에 보여준다.
getchar()	사용자가 키보드로 Enter 를 누를 때까지 입력한 것을 메모리(버퍼)에 모두 저장해놓고(Enter 도 저장됨) 그중에서 한 문자만 꺼낸다.
putchar(문자형 변수)	표준 출력 장치(모니터)에 문자 하나를 출력한다.
putch(문자형 변수)	putchar()와 기능이 동일하다.

표준 입출력 함수

- 문자 입출력 함수 (2/9)

- **getch(), putch() 사용 예**

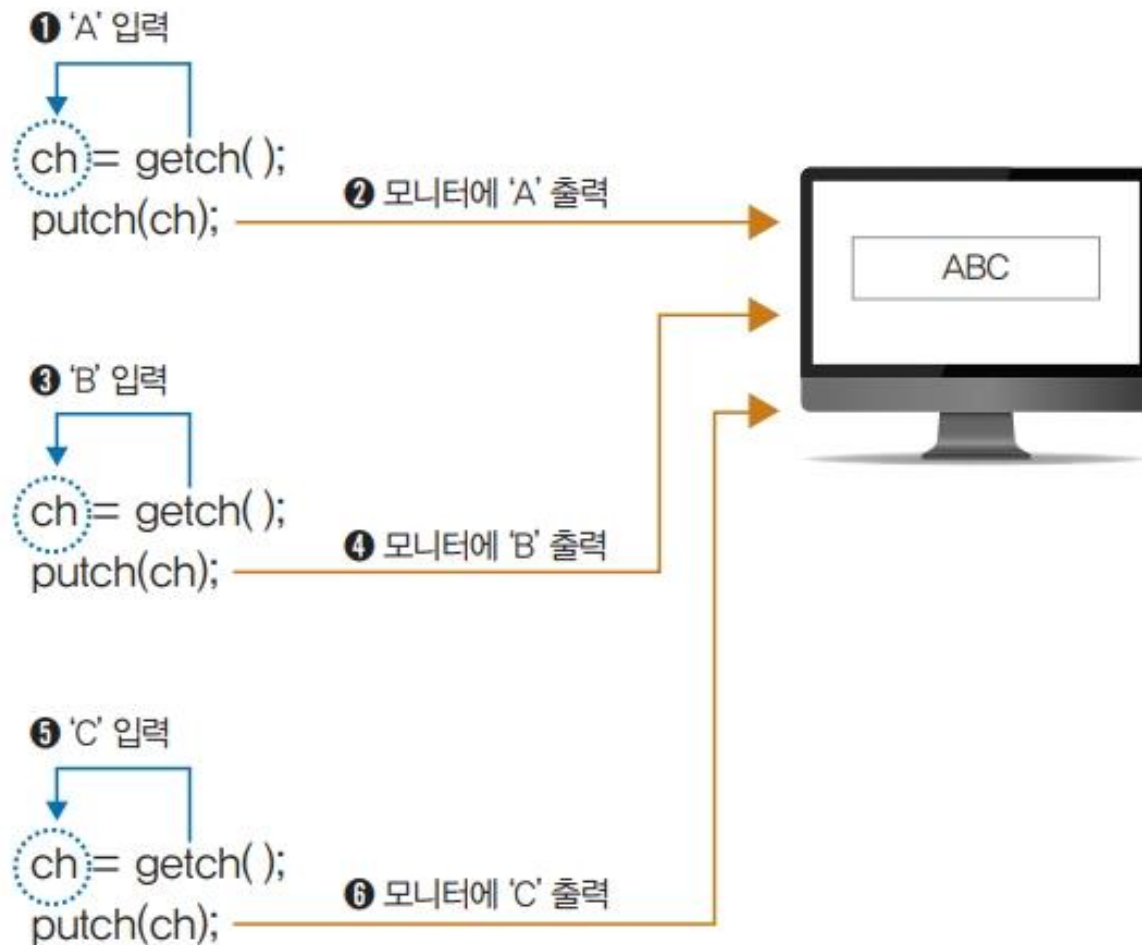


그림 11-2 getch() 함수의 작동

표준 입출력 함수

- 문자 입출력 함수 (3/9)

• getch(), putch() 사용 예제

✓ 비밀번호 검사

응용 11-4 문자 입출력 함수 사용 예 2

11-4.c

```
01 #include <stdio.h>
02 #include <string.h>
03
04 void main( )
05 {
06     char password[5] = "5678";
07     char input[5];
08     int i;
09
10     printf("비밀번호 4글자를 입력하세요 : ");
11     for(i=0; i < 4; i++)
12         input[i] = __1__;
13
14     if(strncmp(password, input, 4) == 0)
15     {
16         printf("\n비밀번호 통과\n");
17     }
```

1 비밀번호를 '5678'로 고정한다.

2 입력받은 비밀번호를 저장하는 문자 배열이다.

3 문자 4개를 입력받는다(입력한 글자는 보이지 않는다).

4 입력한 글자 4개가 비밀번호와 같으면 통과한다.

표준 입출력 함수

- 문자 입출력 함수 (4/9)

• getch(), putch() 사용 예제

```
18  else
19  {
20      printf("\n입력한 비밀번호 ");
21
22      for(i=0; i < 4; i++)
23          __2__(input[i]);
24
25      printf(" 가 틀렸음\n");
26  }
27 }
```

----- 입력한 글자 4개가 비밀번호와 다르면
사용자가 입력한 내용을 출력한다.

putch() getch()

실행 결과

5 비밀번호 4글자를 입력하세요 : --> 입력하는 글자가 안 보임
입력한 비밀번호 1234 가 틀렸음

표준 입출력 함수

- 문자 입출력 함수 (5/9)

• getch()

- ✓ getch() 함수는 입력받은 내용을 모니터에 출력하지 않지만 getche() 함수는 putchar() 함수를 사용하지 않아도 입력한 내용을 바로 모니터에 출력

기본 11-5 문자 입출력 함수 사용 예 3

11-5.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     char ch;
06
07     ch = getche( );    —— 문자 1개를 입력받고 모니터에 보여준다
08
09     ch = getche( );    —— 문자 1개를 입력받고 모니터에 보여준다.
10
11     ch = getche( );    —— 문자 1개를 입력받고 모니터에 보여준다.
12 }
```

실행 결과

ABC

표준 입출력 함수

- 문자 입출력 함수 (6/9)

• getchar()

- ✓ 아래 예제 실행 시 두 글자만 입력하고 Enter를 입력하는 경우와 세 글자를 입력하고 Enter를 입력하는 경우를 비교

기본 11-6 문자 입출력 함수 사용 예 4

11-6.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     char ch;
06
07     printf("문자열을 입력하세요 : ");
```

1

```
08     ch = getchar( );
09     putchar(ch);
```

—— 문자 1개를 입력받는다.

—— 버퍼에서 문자를 읽어 모니터에 출력한다.

2

```
11     ch = getchar( );
12     putchar(ch);
```

—— 문자 1개를 입력받는다.

—— 버퍼에서 문자를 읽어 모니터에 출력한다.

```
13
```

표준 입출력 함수

- 문자 입출력 함수 (7/9)

• getchar()

3

```
14  ch = getchar( );  
15  putchar(ch);  
16 }
```

----- 문자 1개를 입력받는다.
----- 버퍼에서 문자를 읽어 모니터에 출력한다.

실행 결과

문자열을 입력하세요 : AB

AB

--> 빈 줄이 한 줄 출력됨

실행 결과

문자열을 입력하세요 : ABC

ABC --> 빈 줄이 없음

- ✓ 두 문자를 입력하고 Enter를 눌렀을 때는 두 문자가 출력되고 줄이 넘어감
- ✓ 세 문자를 입력하고 Enter를 눌렀을 때는 세 문자만 출력되고 줄이 넘어가지
않음

표준 입출력 함수

- 문자 입출력 함수 (8/9)

• getchar()

- ✓ Enter를 만나기 전까지의 문자들을 버퍼에 저장하고 호출될 때 마다 하나씩 차례대로 반환

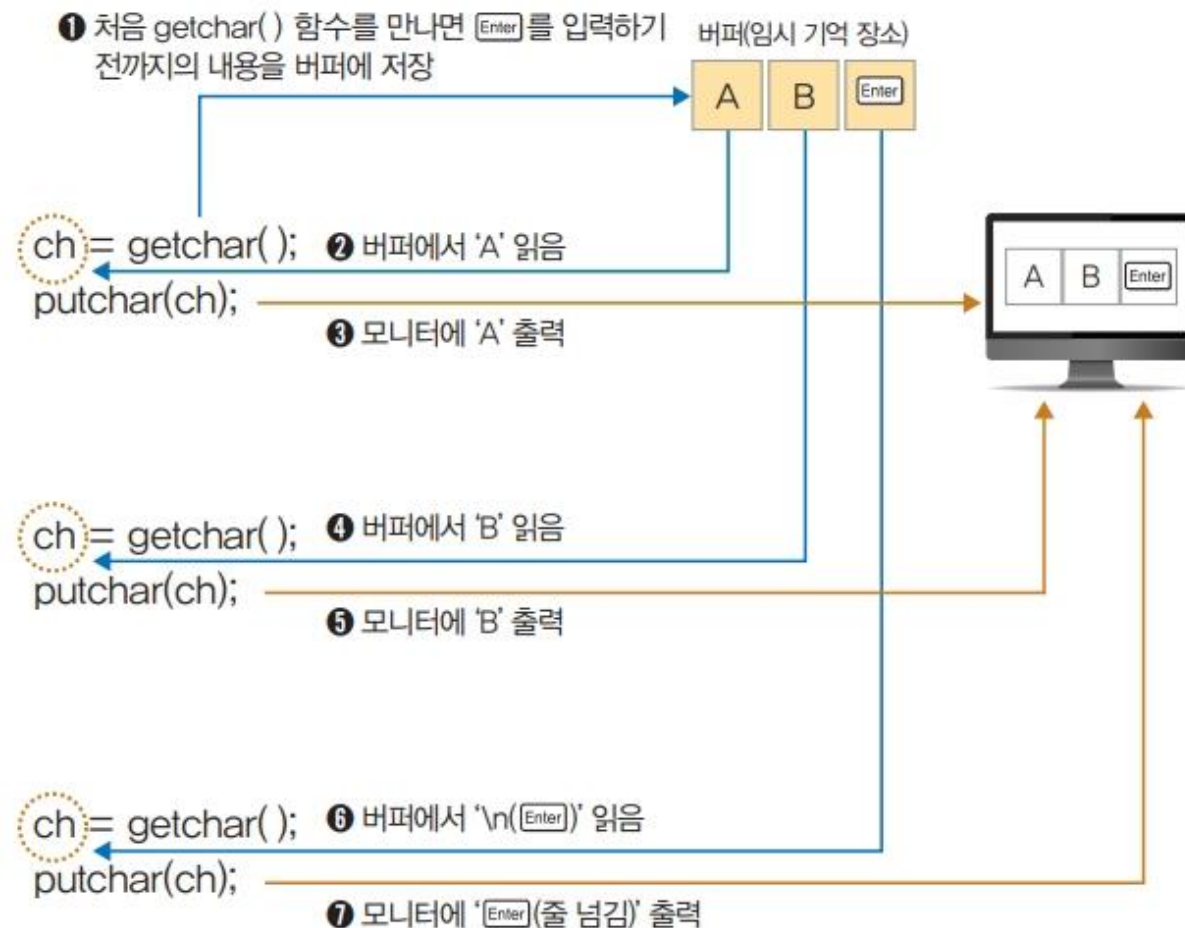


그림 11-3 두 문자를 입력할 때 `getchar()`, `putchar()` 함수의 작동

표준 입출력 함수

- 문자 입출력 함수 (9/9)

- **getchar() 사용 예제**

- ✓ 사용자가 입력한 문자열의 문자들을 차례대로 출력

```
#include <stdio.h>

int main() {
    char c;
    printf("문자열을 입력하세요: ");

    // getchar()를 사용하여 문자를 읽어옴
    while ((c = getchar()) != '\n') {
        // 읽은 문자를 출력
        printf("입력된 문자: %c\n", c);
    }

    return 0;
}
```

```
문자열을 입력하세요: Hello
입력된 문자: H
입력된 문자: e
입력된 문자: l
입력된 문자: l
입력된 문자: o
```

파일 입출력 함수

- 개요

- 표준 입출력과 파일 입출력 함수

- ✓ 사용하는 함수와 입출력 관련 장치가 다름
- ✓ 파일 뿐만 아니라 다양한 입출력 장치들을 사용할 수 있음

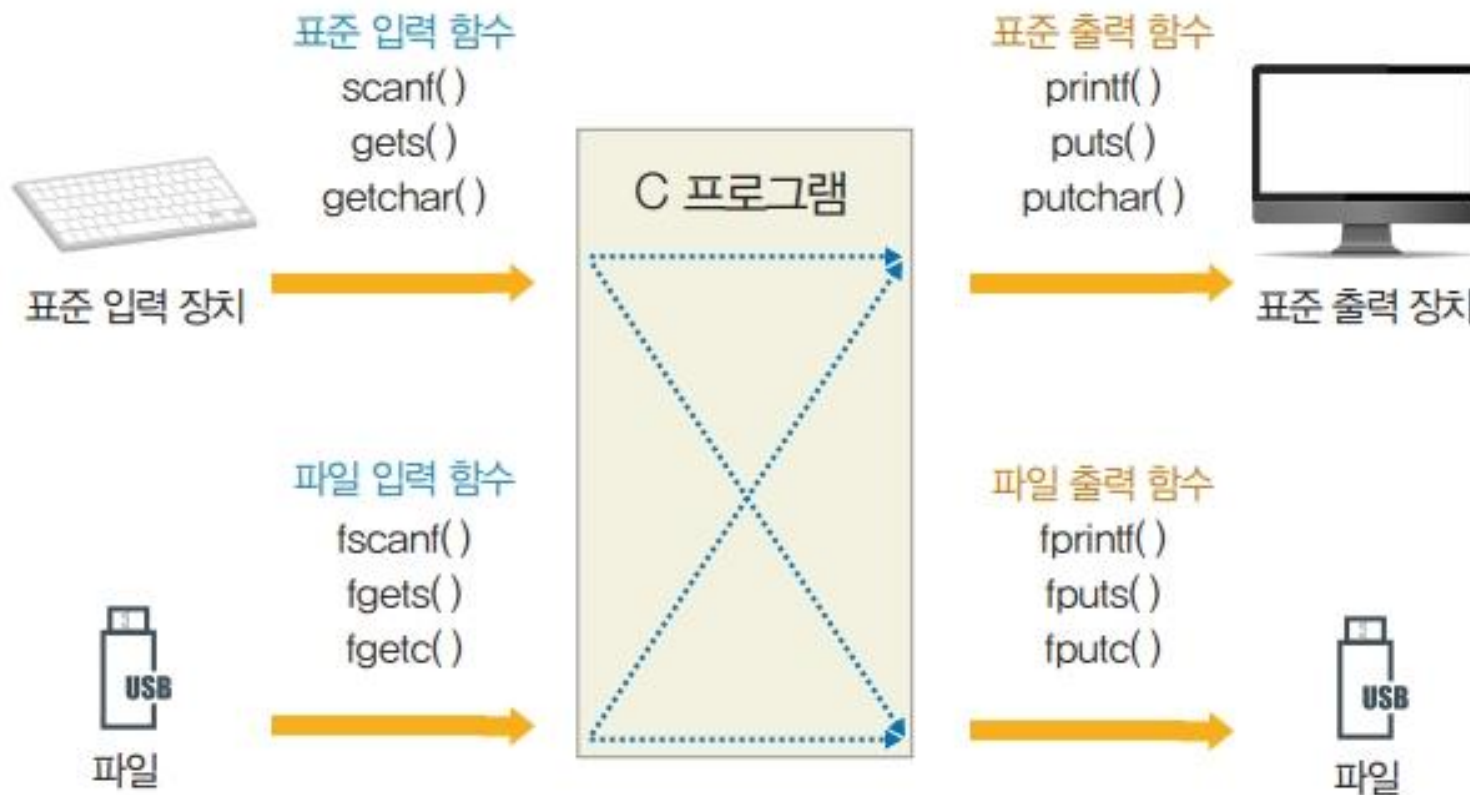


그림 11-4 표준 입출력 함수와 파일 입출력 함수

파일 입출력 함수

- 파일 입출력의 기본 과정

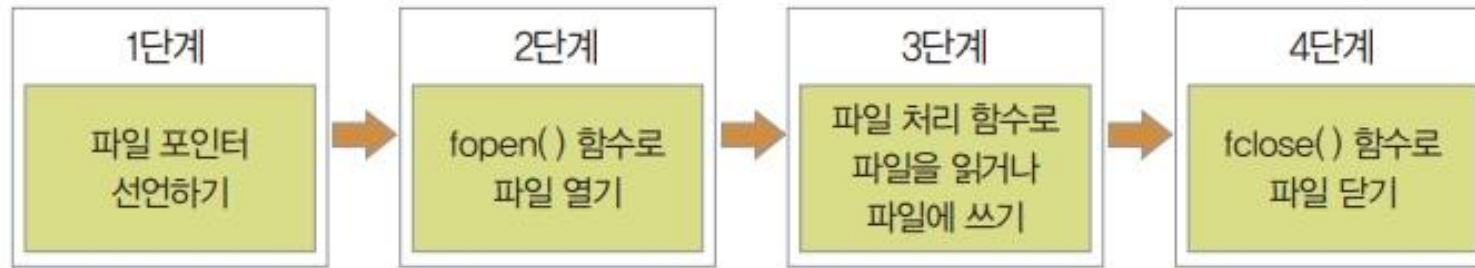


그림 11-5 파일 입출력의 기본 과정

- **1단계 : 파일 포인터 선언**

```
FILE *변수 이름;
```

- **2단계 : fopen() 함수로 파일 열기**

```
변수 이름 = fopen("파일 이름", "열기 모드");
```

- **3단계 : 파일 처리 함수로 파일을 읽거나 파일에 쓰기**

- **4단계 : fclose() 함수로 파일 닫기**

```
fclose(파일 포인터);
```

파일 입출력 함수

- 파일을 이용한 입력 (1/2)

• 파일의 문자열 읽기 : `fgets()` 함수

- ✓ 파일로부터 값을 입력 받을 때 사용
- ✓ 파일 포인터에 지정된 파일에서 문자열을 읽어서 문자 배열에 대입
- ✓ 문자열의 최대 길이는 '읽을 최대 문자수 - 1'

```
fgets(문자 배열, 읽을 최대 문자 수, 파일 포인터);
```



그림 11-6 파일 입력과 표준 출력

파일 입출력 함수

- 파일을 이용한 입력 (2/2)

• fgets() 예제

- ✓ 메모장 실행 → 'File Read Sample' 라는 문구 넣음 → 'C:\temp\data1.txt'로 저장

기본 11-7 파일을 이용한 입력 예 1

11-7.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     char s[20];
06     FILE *rfp;
07
08     rfp=fopen("c:\\temp\\data1.txt", "r");
09
10     fgets(s, 20, rfp);
11
12     printf("파일에서 읽은 문자열 : ");
13     puts(s);
14
15     fclose(rfp);
16 }
```

실행 결과

파일에서 읽은 문자열 : File Read Sample

1

05 char s[20]; ----- 문자 배열을 선언한다.

06 FILE *rfp; ----- 파일 포인터를 선언한다.

2

08 rfp=fopen("c:\\temp\\data1.txt", "r"); ----- 파일 읽기(r) 모드로 연다. 폴더와
09 파일의 경로는 '\'를 2개씩 써야 한다.

3

10 fgets(s, 20, rfp);

13 puts(s); ----- 모니터에 문자열을 출력한다.

파일 입출력 함수

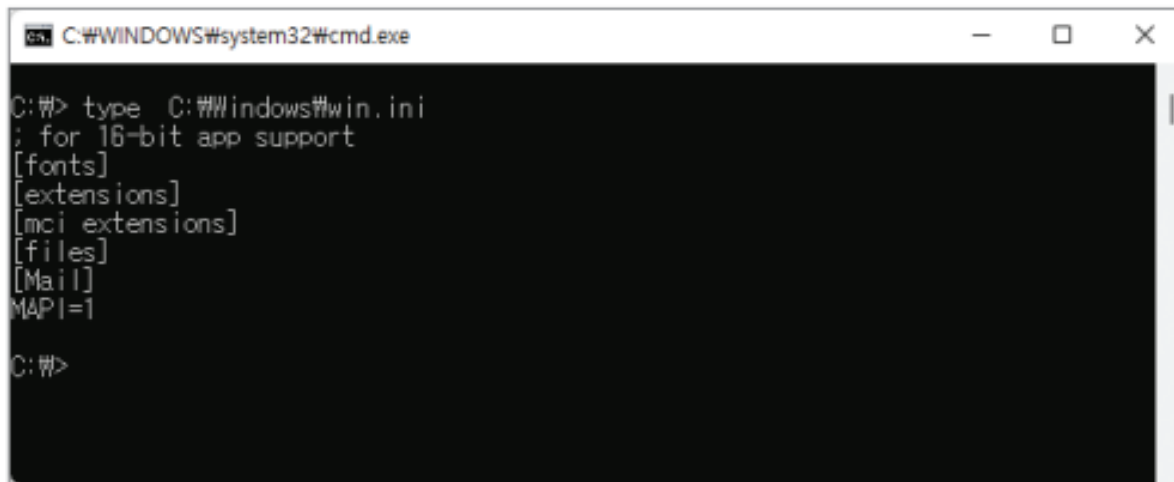
- 도스 명령어 type의 구현 (1/3)

- **type** : 지정한 파일의 내용을 화면에 출력하는 기능

type 파일 이름

- **[시작] → [실행]**을 선택 후 'cmd' 명령을 입력하여 명령 프롬프트를 연다.
- **다음 명령어 입력**

type C:\windows\win.ini



```
C:\WINDOWS\system32\cmd.exe
C:\> type C:\windows\win.ini
; for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAP|=1
C:\>
```


파일 입출력 함수

- 도스 명령어 type의 구현 (2/3)

• type 명령어 프로그램 단계



응용 11-8 파일을 이용한 입력 예 2

11-8.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     char str[200];           —— 한 번에 최대 199자까지 읽을 수 있도록 배열을 선언한다.
06     FILE *rfp;              —— 파일 포인터를 선언한다.
07
08     rfp=fopen("c:\\windows\\win.ini", "r"); —— 읽어올 파일을 연다.
09
10     for( ; ; )              —— 무한 루프이다.
11     {
12         __fgetc__(str, 200, rfp); —— 파일에서 한 줄씩 읽어온다.
13     }
```


파일 입출력 함수

- 도스 명령어 type의 구현 (3/3)

• type 명령어 프로그램 단계

```
14     if( __2__(rfp))  ----- 파일의 끝이라면 for문을 종료한다.
15         break;
16
17     printf("%s", str); ----- 파일의 끝이 아니므로 읽은 내용을 출력한다.
18 }
19
20     __3__(rfp); ----- 파일을 닫는다.
21 }
```

feof() : End Of File
파일의 끝에 도달하면 1을 반환,
그렇지 않으면 0을 반환

fclose 3 feof 2 fgets 1

실행 결과

```
; for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAPI=1
```

파일 입출력 함수

- 서식을 지정하여 파일 읽기 : fscanf() 함수 (1/3)

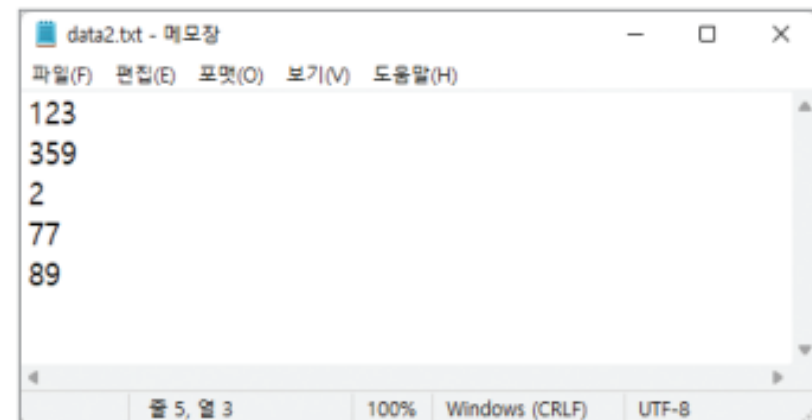
- 파일 포인터를 사용하는 것을 제외하고 **scanfs()**와 사용법이 동일

```
fscanf(파일 포인터, "서식", 입력할 매개변수들...);
```



- **fscanf() 예제**

- ✓ 'C:\temp\data2.txt' 파일에 정수 5줄을 쓰고, fscanf() 함수로 읽어온 후 그 숫자들을 합하는 프로그램을 작성
- ✓ 먼저 5줄의 숫자를 메모장에 적고 'C:\temp\data2.txt'로 저장



파일 입출력 함수

- 서식을 지정하여 파일 읽기 : fscanf() 함수 (2/3)

• fscanf() 예제

기본 11-9 파일을 이용한 입력 예 3

11-9.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     FILE *rfp;          ----- 파일 포인터를 선언한다.
06     int hap=0;          ----- 합계 변수를 선언하고 초기화한다.
07     int in, i;          ----- 읽어올 숫자 변수와 반복을 위한 변수이다.
08
09     rfp=fopen("c:\\temp\\data2.txt", "r"); ----- 파일을 읽기 모드(r)로 읽는다.
10
11     for(i=0; i < 5; i++) ----- 5회 반복하면서 파일 포인터에서 정수를
12     {                      읽어오고 합계를 누적한다.
13         fscanf(rfp, "%d", &in);
14         hap = hap + in;
15     }
```

파일 입출력 함수

- 서식을 지정하여 파일 읽기 : fscanf() 함수 (3/3)

• fscanf() 예제

```
16  
17  printf("합계 ==> %d\n", hap);      —— 합계를 출력한다.  
18  
19  fclose(rfp);                      —— 파일을 닫는다.  
20 }
```

실행 결과

합계 ==> 650

파일 입출력 함수

- 파일을 이용한 출력 (1/2)

• 파일의 문자열 출력 : `fputs()` 함수

- ✓ 파일에서 데이터를 읽어와 화면에 출력하는 대신 파일에 내용을 씀
- ✓ 파일 포인터에 지정된 파일에 문자열을 출력

```
fputs(출력할 데이터, 파일 포인터);
```

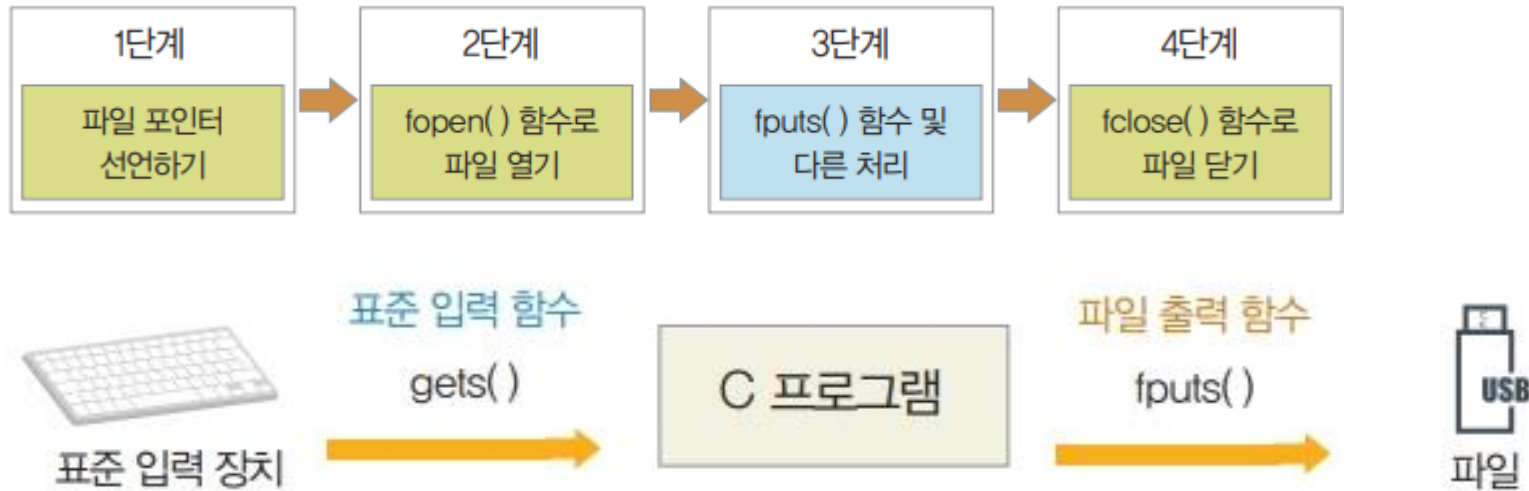


그림 11-7 표준 입력과 파일 출력

파일 입출력 함수

- 파일을 이용한 출력 (2/2)

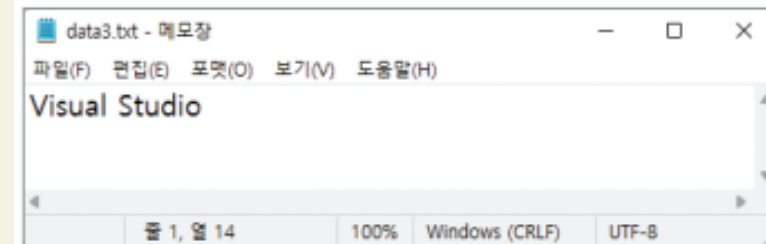
• fputs() 함수 예제

기본 11-10 파일을 이용한 출력 예 1

11-10.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     char s[20];
06     FILE *wfp;
07
08     wfp=fopen("c:\\temp\\data3.txt", "w");
09
10     printf("문자열을 입력(최대 19자) : ");
11     gets(s);
12
13     fputs(s, wfp);
14
15     fclose(wfp);
16 }
```

실행 결과 ▼ C:\temp\data3.txt의 내용



1 08 wfp=fopen("c:\\temp\\data3.txt", "w"); ----- 파일을 연다('w'는 쓰기 모드를 뜻한다).

2 10 printf("문자열을 입력(최대 19자) : ");
11 gets(s); ----- 최대 19자까지 입력할 수 있다.

3 12
13 fputs(s, wfp); ----- 입력된 내용을 파일에 쓴다.

파일 입출력 함수

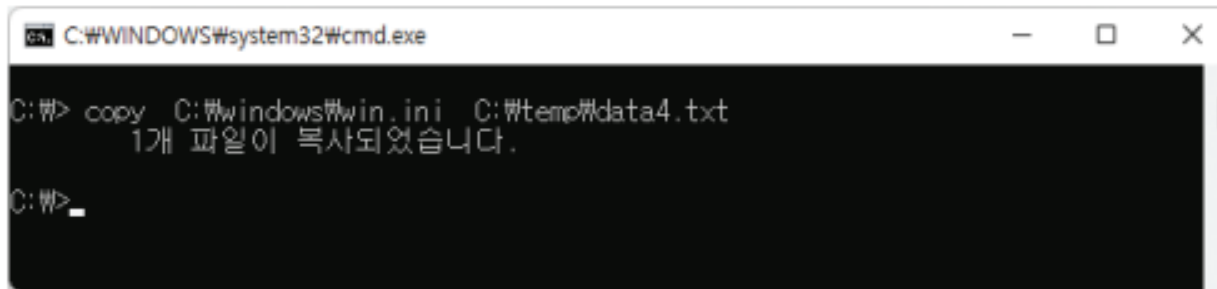
- 도스 명령어 copy의 구현 (1/3)

- **copy** : 주어진 파일을 복사하여 똑같은 파일을 하나 더 만드는 명령어

```
copy 소스_파일 타겟_파일
```

- **[시작] → [실행]**을 선택한 후 'cmd' 명령을 입력하여 명령 프롬프트 창을 연다

```
copy C:\windows\win.ini C:\temp\data4.txt
```



- **파일의 복사 과정**

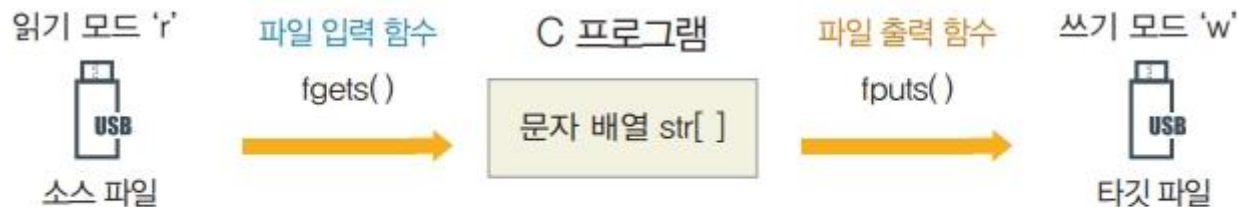


그림 11-8 파일을 이용한 출력

파일 입출력 함수

- 도스 명령어 copy의 구현 (2/3)

구현 예제

응용 11-11 파일을 이용한 출력 예 2

11-11.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
```

1 05 char str[200]; ----- 한 줄에 최대 199자까지 쓸 수 있다.
06 FILE *rfp; ----- 읽기용, 쓰기용 파일 포인터를 허용한다.
07 FILE *wfp;

2 08
09 rfp=fopen("c:\\windows\\win.ini", "r"); ----- 읽기 모드와 쓰기 모드로 파일을 연다.
10 wfp=fopen("c:\\temp\\data5.txt", "w");

11
12 for(; ;) ----- 무한 루프이다.

3 13 {
14 1__(str, 200, rfp); ----- 읽기용 파일에서 한 줄을 읽는다. 최대
15 199자까지 읽을 수 있다.
16 if(feof(rfp)) ----- 읽기용 파일의 끝을 만나 for문을
17 break; 빠져나간다.

파일 입출력 함수

- 도스 명령어 copy의 구현 (3/3)

• 구현 예제

4 18
19 2 ----- 쓰기용 파일에 한 줄을 쓴다.

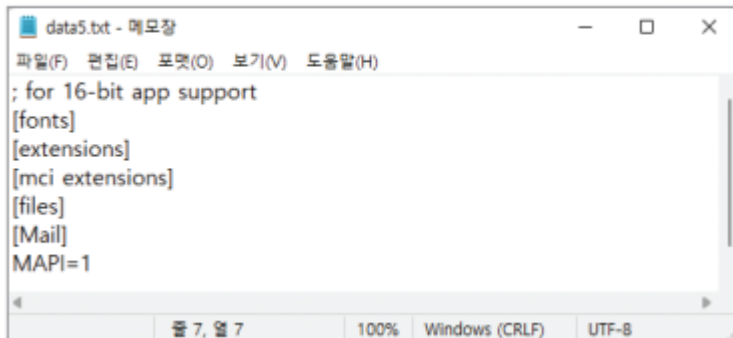
5 21
22 fclose(rfp);
23 fclose(wfp);
24 }

실행 결과

실행 결과

아무것도 출력되지 않음.

실행 결과 ▼ C:\temp\data5.txt의 내용



파일 입출력 함수

- 서식을 지정하여 파일 출력 : fprintf() 함수 (1/2)

- **파일에 숫자를 출력할때는 서식을 지정할 수 있는 fprintf() 함수를 사용하는 것이 편리함**

```
fprintf(파일 포인터, "서식", 출력할 매개변수들 ...);
```

파일 입출력 함수

- 서식을 지정하여 파일 출력 : fprintf() 함수 (2/2)

• fprintf() 함수 예제

기본 11-12 파일을 이용한 출력 예 3

11-12.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     FILE *wfp;
06     int hap=0;
07     int in, i;
08
09     wfp=fopen("c:\\temp\\data7.txt", "w");
10
11     for(i=0; i < 5; i++)
12     {
13         printf(" 숫자 %d : ", i+1);
14         scanf("%d", &in);
15         hap = hap + in;
16     }
17
18     fprintf(wfp, "합계 ==> : %d\n", hap);
19
20     fclose(wfp);
21 }
```

파일을 쓰기 모드로 연다.

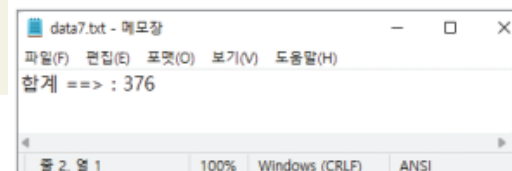
5회 반복하면서 키보드에서 입력받은 숫자의 합계를 누적한다.

합계를 파일에 쓴다.

실행 결과

숫자 1 : 10
숫자 2 : 222
숫자 3 : 35
숫자 4 : 68
숫자 5 : 41

실행 결과 ▼ C:\temp\data7.txt의 내용



실습

[실습 1] 사용자 입력 문자열을 반대로 파일에 저장하기

예제 설명 입력한 순서의 반대로 그리고 각 행의 문자도 반대 순서로 출력하는 프로그램이다([응용 12-7] 활용).

실행 결과

1 번째 문자열 : IT CookBook
2 번째 문자열 : Basic C
3 번째 문자열 : Programming

-- 입력과 반대로 출력(포인터) : 글자 순서도 거꾸로 --
3 :gnimmargorP
2 :C cisaB
1 :kooBkooC TI

 **output1.txt에 저장**

1. 5주차 실습 3번을 참고하여 구현

[실습 2] 파일에서 읽어온 문자열을 반대로 파일에 저장

예제 설명 미리 만들어둔 'in.txt' 파일의 내용을 읽어와 'out.txt' 파일에 쓰는데 각 행의 문자를 반대 순서로 저장하는 프로그램이다(반드시 'in.txt' 파일의 마지막 행에서 `Enter`를 누르고 저장한다).

실행 결과

```
Visual  
Studio Community  
Basic-C  
Study
```

```
lausiV  
ytinummoC oidutS  
C-cisaB  
ydutS
```

1. 실습 1의 결과물을 확장하여 구현
2. in.txt는 직접 만들 것
3. 출력 결과는 output2.txt에 기록

[실습 3] 단어 오름차순, 내림차순 정렬 프로그램

예제 설명

파일을 읽고, 오름차순 내림차순으로 각각 정렬 한 후 각각의 파일에 결과를 저장하시오.
[출력 양식은 아래를 따릅니다.]

실행 결과

원본 데이터 (data.txt)

```
≡ data.txt
1 Grape,Horse,Ice Cream,Juice,Kiwi,
2 Result,Apple,Cat,Dog,Elephant,Fish,sort,Lion,
3 Mango,New,Pear,Banana,
4 Quatro,Orange
5
```

내림차순 결과 (Result_Descending.txt)

```
≡ Result_Descending.txt
1 =====sorting Result=====
2 [ sort]
3 [ Result]
4 [ Quatro]
5 [ Pear]
6 [ Orange]
7 [ New]
8 [ Mango]
9 [ Lion]
10 [ Kiwi]
11 [ Juice]
12 [ Ice Cream]
13 [ Horse]
14 [ Grape]
15 [ Fish]
16 [ Elephant]
17 [ Dog]
18 [ Cat]
19 [ Banana]
20 [ Apple]
21
```

오름차순 결과 (Result_Ascending.txt)

```
≡ Result_Ascending.txt
1 =====sorting Result=====
2 [ Apple]
3 [ Banana]
4 [ Cat]
5 [ Dog]
6 [ Elephant]
7 [ Fish]
8 [ Grape]
9 [ Horse]
10 [ Ice Cream]
11 [ Juice]
12 [ Kiwi]
13 [ Lion]
14 [ Mango]
15 [ New]
16 [ Orange]
17 [ Pear]
18 [ Quatro]
19 [ Result]
20 [ sort]
21
```

[실습 3] 단어 오름차순, 내림차순 정렬 프로그램

조건

1. 포인터 배열 2개를 선언하여 활용하세요
 - a. 파일을 읽어 저장하는 포인터 배열_A
 - i. 파일을 읽어 내용을 저장 할 포인터 배열의 행은 5로 설정한다.
 - b. 저장된 데이터를 단어 단위로 잘라(파싱) 저장 할 포인터 배열_B
 - i. 파싱한 내용을 저장 할 포인터 배열의 행은 20으로 정한다
2. strtok 등 파싱 함수는 활용하지 마세요
3. 총 네개의 함수(sort_Ascending, sort_Descending, write_result_to_file) 함수를 작성하고 구현하세요

상세 설명

1. Main
 - a. data.txt 파일을 읽기 형식으로 불러옴
 - b. getline, getc 등 자유로운 방식으로 파일 내용에 접근하여 포인터배열_A에 저장
 - c. 단어 외 특수문자에 대한 제거를 통해, 단어만을 추출하여 단어를 포인터배열_B에 저장
 - d. 저장한 단어를 오름차순으로 함수 호출을 통해 포인터 배열_B에 정렬하여 저장
 - e. 정렬 된 내용을 함수 호출을 통해 결과 저장 파일(Result_Ascending.txt)에 작성하세요.
 - f. 오름차순으로 정렬된 포인터 배열 내부의 데이터를 함수 호출을 통해 내림차순으로 정렬하여, 포인터 배열_B에 저장
 - g. 내림차순으로 정렬 된 내용을 함수 호출을 통해 결과 저장 파일(Result_Descending.txt)에 저장하세요
2. sort_Ascending
 - a. 파싱 된 데이터를 저장한 포인터배열_B의 데이터를 오름차순으로 정렬하여, 포인터배열_B에 저장하는 함수
3. sort_Descending
 - a. 오름차순으로 정렬된 데이터를 저장한 포인터배열_B의 데이터를 내림차순으로 정렬하여, 포인터배열_B에 저장하는 함수
4. write_result_to_file
 - a. 결과를 파일에 저장하는 함수

Q & A