

Chapter 11

표준 입출력과 파일 입출



목차

1. 표준 입출력 함수
2. 파일 입출력 함수

01

표준 입출력 함수

1. 표준 입출력 함수

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

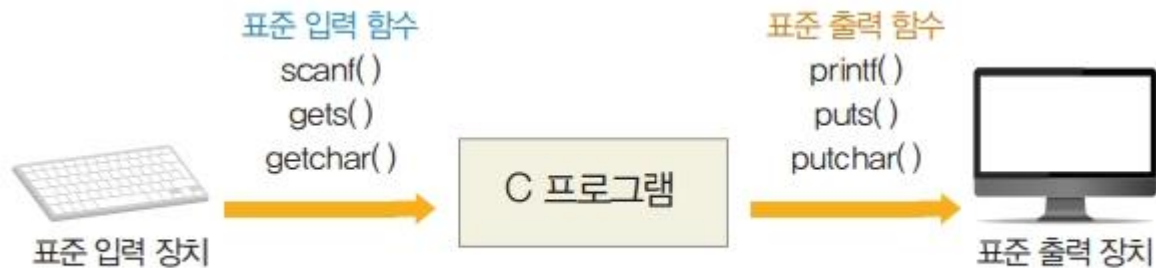


그림 11-1 표준 입출력의 개념

여기서 잠깐 보안이 강화된 입력 함수

- `scanf()`와 `gets()` 함수는 자체적인 보안에 취약하다는 문제가 있음
- 마이크로소프트에서는 이런 문제가 해결된 `scanf_s()`와 `gets_s()` 함수의 사용을 권장
- 이 책에서는 대부분의 C/C++ 컴파일러에서 공통적으로 사용할 수 있도록 `scanf()`와 `gets()` 함수를 사용

1. 표준 입출력 함수

1. 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

1. 표준 입출력 함수

1. 서식화된 입출력 함수

기본 11-1 서식화된 입출력 함수 사용 예

11-1.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     int a;           — 정수형을 선언한다.
06     float b;         — 실수형을 선언한다.
07     char ch;         — 문자형을 선언한다.
08     char s[20];      — 문자 배열을 선언한다.
09
10     printf("정수를 입력 : ");
11     scanf("%d", &a); — 정수를 입력한다.
12     printf("실수를 입력 : ");
13     scanf("%f", &b); — 실수를 입력한다.
14     printf("문자를 입력 : ");
15     scanf(" %c", &ch); — 문자를 입력한다. 13행의 Enter 를 무시하기 위해
                        %c 앞에 공백이 필요하다.
16     printf("문자열을 입력 : ");
17     scanf("%s", s); — 문자열을 입력받는다. 배열 이름 s는 그 자체가
                        주소이므로 &를 사용하지 않는다.
18
```

1. 표준 입출력 함수

1. 서식화된 입출력 함수

```
19  printf("\n정수의 10진수 ==> %d\n", a);
20  printf("정수의 16진수 ==> %X\n", a);
21  printf("정수의 8진수 ==> %o\n", a);
22  printf("실수 ==> %10.3f\n", b);
23  printf("실수(공학용) ==> %e\n", b);
24  printf("문자 ==> %c\n", ch);
25  printf("문자열 ==> %s\n", s);
26 }
```

정수를 10진수, 16진수, 8진수로 출력한다.

실수를 일반 방식 및 공학용으로 출력한다.

문자 및 문자열로 출력한다.

실행 결과

정수를 입력 : 1234
실수를 입력 : 100.12345
문자를 입력 : K
문자열을 입력 : IT_CookBook

정수의 10진수 ==> 1234
정수의 16진수 ==> 4D2
정수의 8진수 ==> 2322
실수 ==> 100.123
실수(공학용) ==> 1.001235e+02
문자 ==> K
문자열 ==> IT_CookBook

1. 표준 입출력 함수

2. 문자열 입출력 함수

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

표 11-3 문자열 입출력 함수 종류

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

- 문자열만 입출력할 경우에는 `printf()`, `scanf_s()` 함수보다는 `gets_s()`, `puts()` 함수를 사용하는 것이 처리 속도가 더 빠름.

1. 표준 입출력 함수

2. 문자열 입출력 함수

기본 11-2 문자열 입출력 함수 사용 예

11-2.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     char s[20];          —— 문자 배열을 선언한다.
06
07     printf("문자열을 입력 : ");
08     gets(s);             —— 문자열을 입력한다. 최대 19자까지 입력할 수 있다.
09
10     puts(s);             —— 문자열을 출력한다.
11 }
```

실행 결과

문자열을 입력 : IT CookBook

IT CookBook

1. 표준 입출력 함수

3. 문자 입출력 함수

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

1. 표준 입출력 함수

3. 문자 입출력 함수

기본 11-3 문자 입출력 함수 사용 예 1

11-3.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05     char ch;
06
07     ch = getch( ); ----- 문자 1개를 입출력한다.
08     putchar(ch);
09
10     ch = getch( ); ----- 문자 1개를 입출력한다.
11     putchar(ch);
12
13     ch = getch( ); ----- 문자 1개를 입출력한다.
14     putchar(ch);
15 }
```

실행 결과

ABC

1. 표준 입출력 함수

3. 문자 입출력 함수

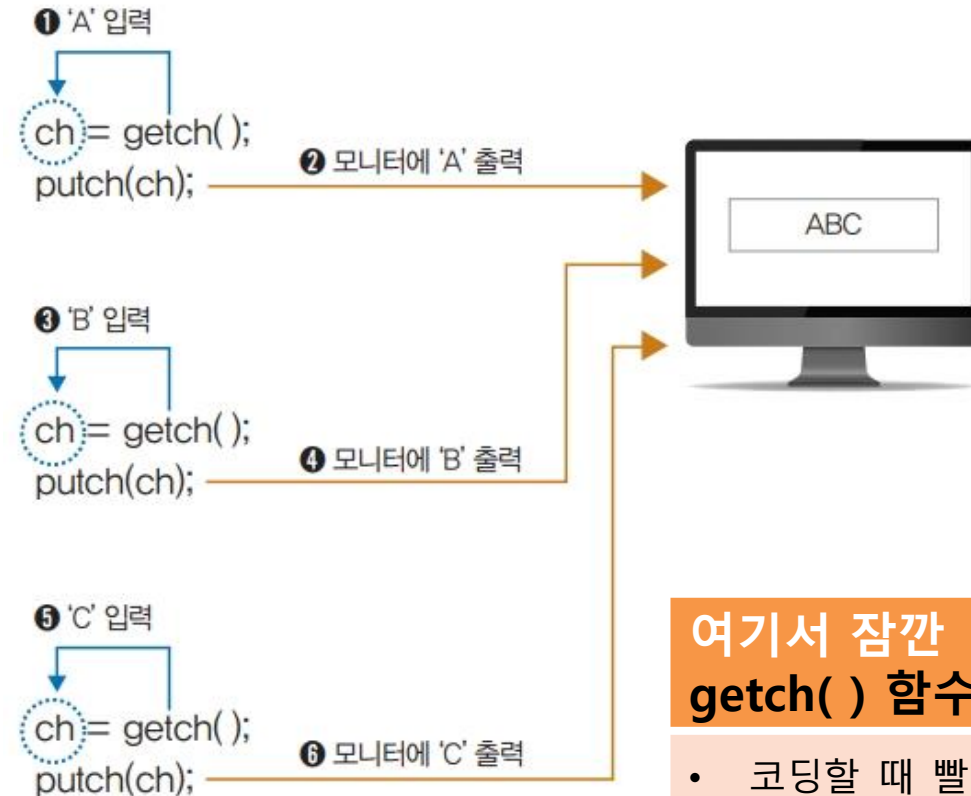


그림 11-2 `getch()` 함수의 작동

여기서 잠깐

`getch()` 함수와 `getche()` 함수의 사용 유무

- 코딩할 때 빨간 줄 오류 표시가 나는 것이 거슬린다면 `_getch()`, `_getche()` 함수를 사용하고 추가로 `#include`를 쓰면 이상 없음

1. 표준 입출력 함수

3. 문자 입출력 함수

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     }
```

----- 비밀번호를 '5678'로 고정한다.

----- 입력받은 비밀번호를 저장하는 문자 배열이다.

----- 문자 4개를 입력받는다(입력한 글자는 보이지 않는다).

----- 입력한 글자 4개가 비밀번호와 같으면 통과한다.

1. 표준 입출력 함수

3. 문자 입출력 함수

```
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.

putc () ()

실행 결과

비밀번호 4글자를 입력하세요 :
입력한 비밀번호 1234 가 틀렸음

--> 입력하는 글자가 안 보임

1. 표준 입출력 함수

3. 문자 입출력 함수

▪ getche() 함수

- getch() 함수는 입력받은 내용을 모니터에 출력하지 않지만 getche() 함수는 putch() 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

1. 표준 입출력 함수

3. 문자 입출력 함수

▪ getchar() 함수

- [기본 11-6]을 실행하여 두 글자를 입력하고 Enter를 눌러본 후 세 글자를 입력하고 Enter를 다시 눌러 실행

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 : ");
08     ch = getchar( );      —— 문자 1개를 입력받는다.
09     putchar(ch);          —— 버퍼에서 문자를 읽어 모니터에 출력한다.
10
11     ch = getchar( );      —— 문자 1개를 입력받는다.
12     putchar(ch);          —— 버퍼에서 문자를 읽어 모니터에 출력한다.
13
```


1. 표준 입출력 함수

3. 문자 입출력 함수

▪ getchar() 함수

```
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

1. 표준 입출력 함수

3. 문자 입출력 함수

▪ getchar() 함수

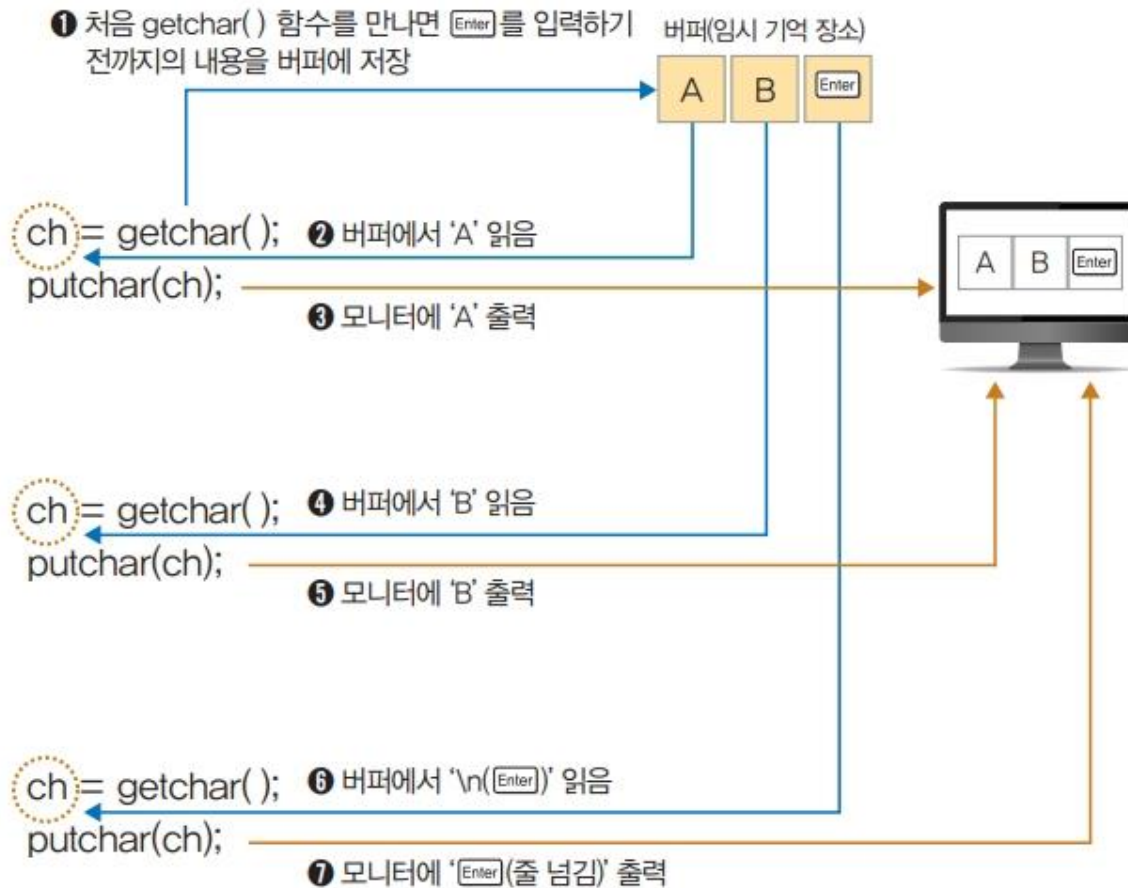


그림 11-3 두 문자를 입력할 때 getchar(), putchar() 함수의 작동

02

파일 입출력 함수

2. 파일 입출력 함수

- 표준 입출력과 파일 입출력 함수
 - 사용하는 함수와 입출력 관련 장치가 다름

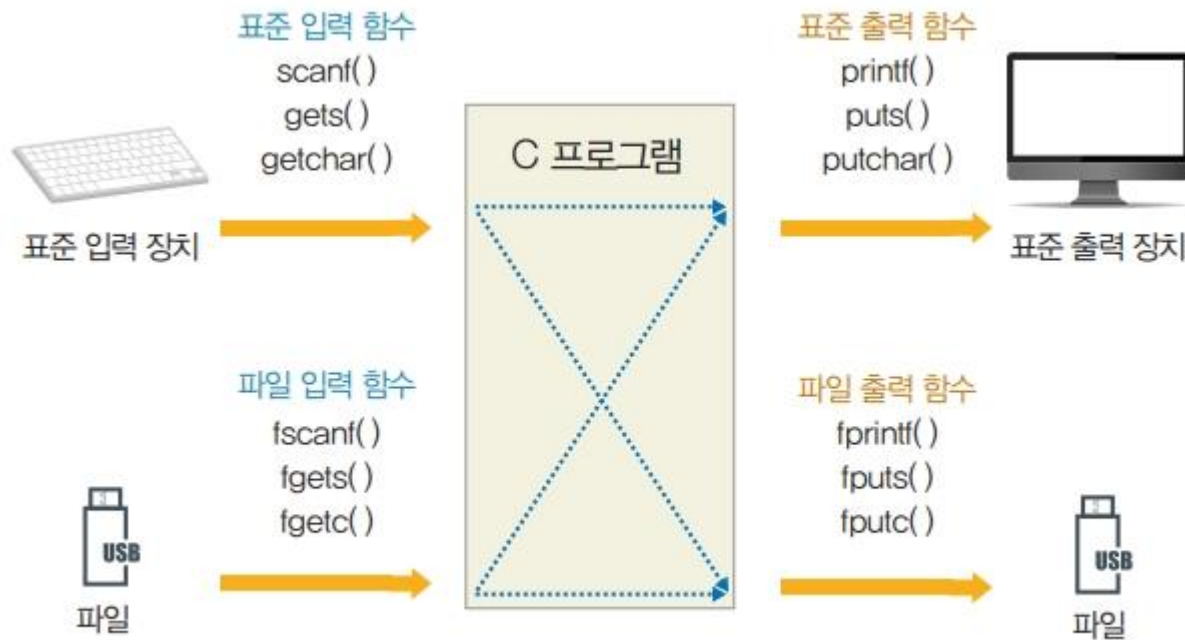


그림 11-4 표준 입출력 함수와 파일 입출력 함수

2. 파일 입출력 함수

1. 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);
```

2. 파일 입출력 함수

2. 파일을 이용한 입력

- 파일의 문자열 읽기 : `fgets()` 함수

- 파일로부터 값을 입력받을 때 사용하며, 파일 포인터에 지정된 파일에서 문자열을 읽어서 문자 배열에 대입함. 문자열의 최대 길이는 '읽을 최대 문자수 -1'

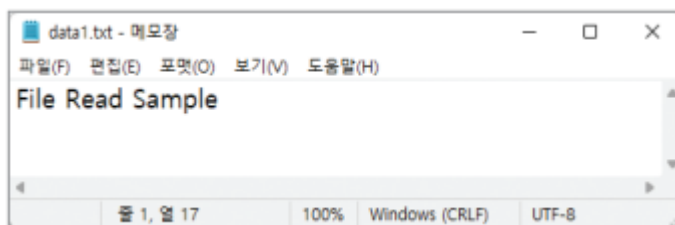
```
fgets(문자 배열, 읽을 최대 문자 수, 파일 포인터);
```



그림 11-6 파일 입력과 표준 출력

- 파일을 통해 데이터를 입력한 후 이를 모니터에 출력하는 프로그램을 작성해보고자 .

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



2. 파일 입출력 함수

2. 파일을 이용한 입력

- 파일의 문자열 읽기 : `fgets()` 함수

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];           ----- 문자 배열을 선언한다.
06     FILE *rfp;           ----- 파일 포인터를 선언한다.
07
08     rfp=fopen("c:\\temp\\data1.txt", "r"); ----- 파일 읽기(r) 모드로 연다. 폴더와
09                                     파일의 경로는 '\'를 2개씩 써야 한다.
10     fgets(s, 20, rfp);
11
12     printf("파일에서 읽은 문자열 : ");
13     puts(s);             ----- 모니터에 문자열을 출력한다.
14
15     fclose(rfp);
16 }
```

실행 결과

파일에서 읽은 문자열 : File Read Sample

2. 파일 입출력 함수

- 도스 명령어 **type**의 구현

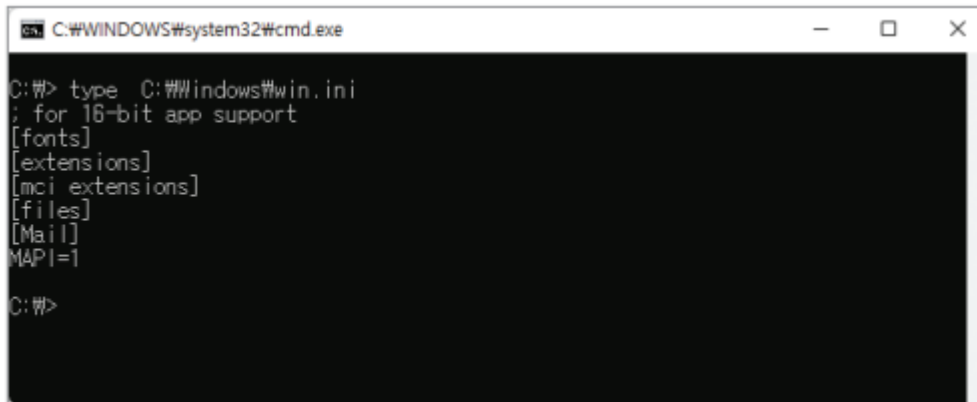
- type : 지정한 파일의 내용을 화면에 출력하는 기능

```
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]
MAPI=1
C:\>
```


2. 파일 입출력 함수

- '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"); —— 읽어들 파일을 연다.
09
10     for( ; ; )              —— 무한 루프이다.
11     {
12         1 (str, 200, rfp); —— 파일에서 한 줄씩 읽어온다.
13
```

2. 파일 입출력 함수

```
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 }
```

fclose 3 feof 2 fgets 1

실행 결과

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

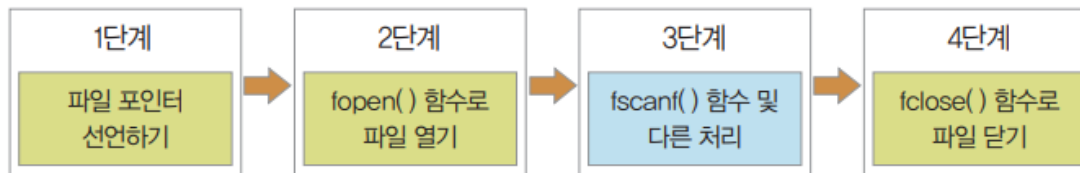
2. 파일 입출력 함수

▪ 서식을 지정하여 파일 읽기 : fscanf() 함수

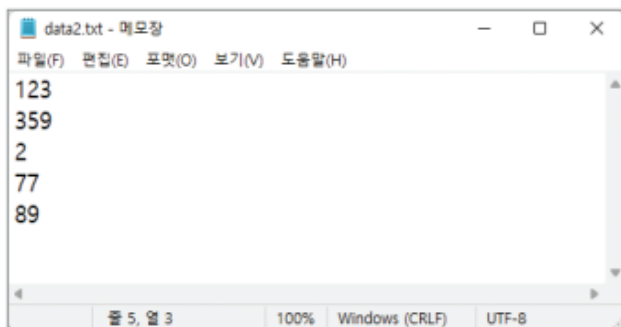
- 파일 포인터를 사용하는 것을 제외하고 scanf()와 사용법이 동일

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

- 'C:\temp\data2.txt' 파일에 정수 5줄을 쓰고, fscanf() 함수로 읽어온 후 그 숫자들을 합하는 프로그램을 작성 과정



먼저 5줄의 숫자를 메모장에 적고 'C:\temp\data2.txt'로 저장함.



2. 파일 입출력 함수

- 서식을 지정하여 파일 읽기 : fscanf() 함수

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     }
```

2. 파일 입출력 함수

- 서식을 지정하여 파일 읽기 : fscanf() 함수

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

실행 결과

합계 ==> 650

- [기본 11-9]에서는 6행에서 숫자의 합계를 넣을 hap 변수를 선언하고 0으로 초기화
- 7행에서는 파일로부터 숫자를 읽어들이는 변수 in을 선언하고 9행에서 data2.txt를 읽기 모드로 열기
- 13행에서 fscanf() 함수를 사용하여 in에 정숫값을 읽어들이는 과정을 5회 반복
- 14행의 hap에는 읽어들이는 숫자를 누적하며, 17행에서 누적된 합계를 출력하고 19행에서 파일을 닫기

2. 파일 입출력 함수

3. 파일을 이용한 출력

- 파일의 문자열 출력 : `fputs()` 함수

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

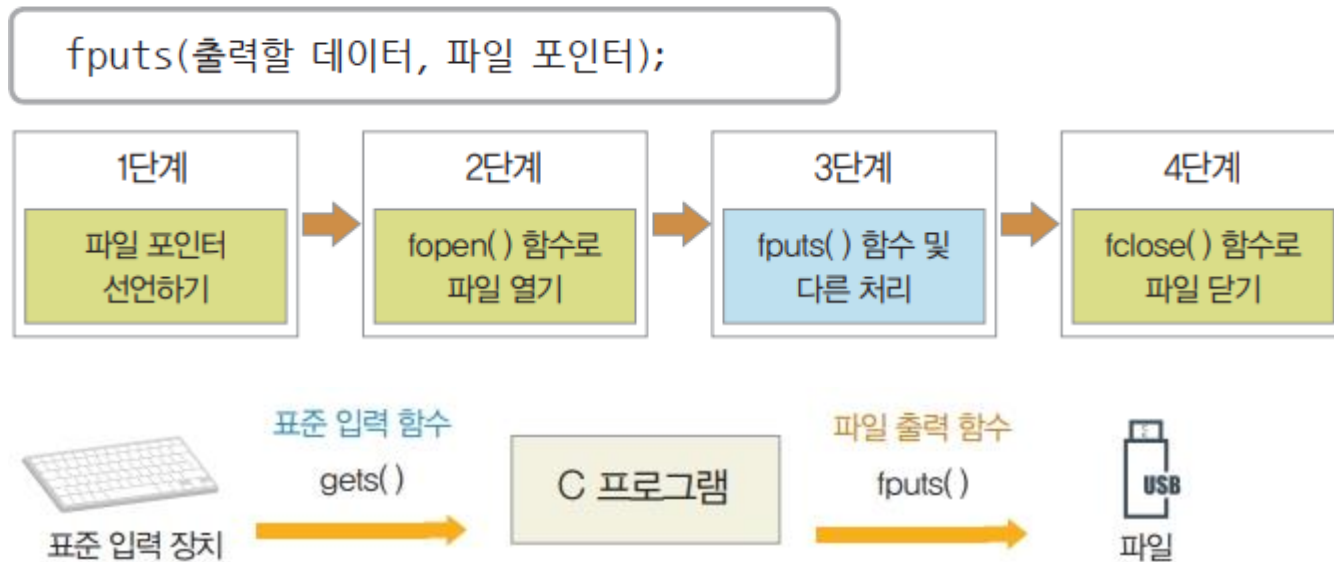


그림 11-7 표준 입력과 파일 출력

2. 파일 입출력 함수

3. 파일을 이용한 출력

- 파일의 문자열 출력 : `fputs()` 함수

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");    ----- 파일을 연다('w'는 쓰기 모드를 뜻한다).
09
10     printf("문자열을 입력(최대 19자) : ");
11     gets(s);    ----- 최대 19자까지 입력할 수 있다.
12
13     fputs(s, wfp);    ----- 입력된 내용을 파일에 쓴다.
14
15     fclose(wfp);
16 }
```

실행 결과

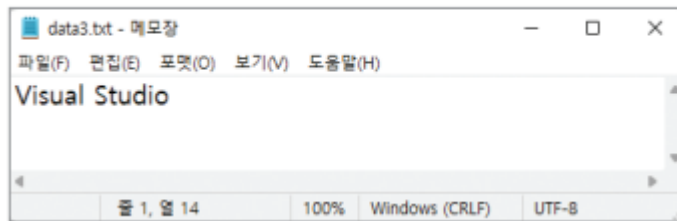
문자열을 입력(최대 19자) : Visual Studio

2. 파일 입출력 함수

3. 파일을 이용한 출력

- 파일의 문자열 출력 : `fputs()` 함수

실행 결과 ▼ C:\temp\data3.txt의 내용



- [기본 11-10]의 6행에서 쓰기용 파일 포인터를 `wfp`로 선언하고 8행에서 `C:\temp\data3.txt` 파일을 쓰기 모드로 열기
- 11행에서는 키보드로 입력받은 문자열을 배열 `s`에 저장하고 13행에 서는 `fputs` 함수를 사용하여 파일에 쓰기
- 15행에서 파일을 닫기
- 프로그램을 실행한 후 `data3.txt` 파일을 열어보면 방금 입력한 내용이 들어 있음

2. 파일 입출력 함수

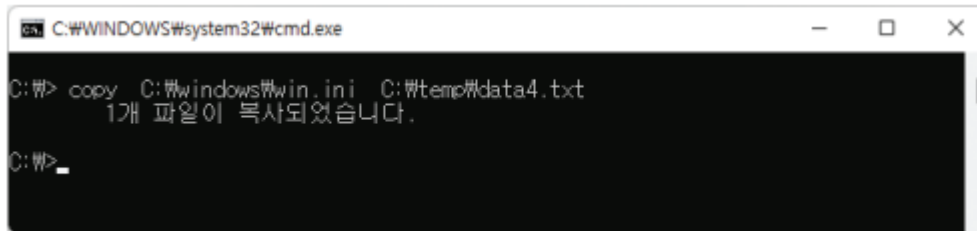
■ 도스 명령어 copy의 구현

- 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

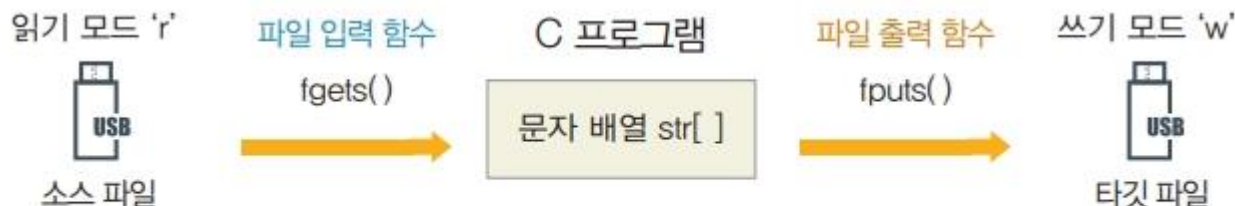


그림 11-8 파일을 이용한 출력

2. 파일 입출력 함수

■ 도스 명령어 copy의 구현

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 {
05     char str[200];           — 한 줄에 최대 199자까지 쓸 수 있다.
06     FILE *rfp;               — 읽기용, 쓰기용 파일 포인터를 허용한다.
07     FILE *wfp;
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.
13     {
14         __f__(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.
```

2. 파일 입출력 함수

- 도스 명령어 copy의 구현

```
18
```

```
19
```

```
2
```

쓰기용 파일에 한 줄을 쓴다.

```
20
```

```
}
```

```
21
```

```
22 fclose(rfp);
```

```
23 fclose(wfp);
```

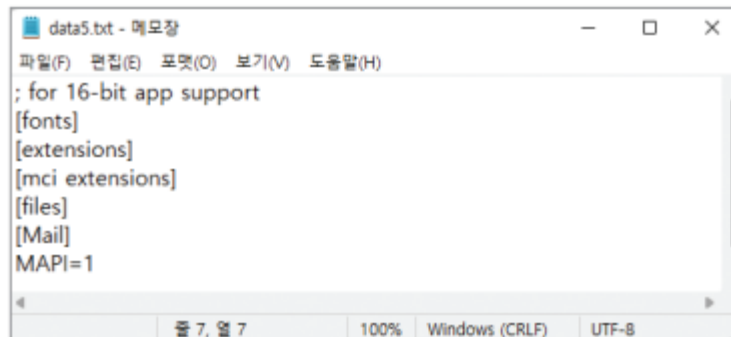
```
24 }
```

return fgetc(str, wfp);

실행 결과

아무것도 출력되지 않음.

실행 결과 ▼ C:\temp\data5.txt의 내용



2. 파일 입출력 함수

- 서식을 지정하여 파일 출력 : `fprintf()` 함수
 - 파일에 숫자를 출력할 때는 서식을 지정할 수 있는 `fprintf()` 함수를 사용하는 것이 편리함

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

2. 파일 입출력 함수

- 서식을 지정하여 파일 출력 : fprintf() 함수

기본 11-12 파일을 이용한 출력 예 3

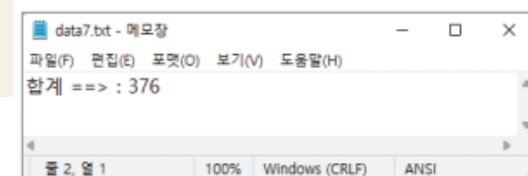
11-12.c

```
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");    —— 파일을 쓰기 모드로 연다.
10
11     for(i=0; i < 5; i++)                    —— 5회 반복하면서 키보드에서 입력받은
12     {                                       숫자의 합계를 누적한다.
13         printf(" 숫자 %d : ", i+1);
14         scanf("%d", &in);
15         hap = hap + in;
16     }
17
18     fprintf(wfp, "합계 ==> : %d\n", hap);    —— 합계를 파일에 쓴다.
19
20     fclose(wfp);
21 }
```

실행 결과

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

실행 결과 ▼ C:\temp\data7.txt의 내용



*

예제 모음

[예제모음 30] 구구단을 파일에 출력

예제 설명 [예제모음 14]의 내용을 모니터가 아닌 'gugu.txt' 파일에 쓰는 프로그램이다.

실행 결과


#제2단# #제3단# #제4단# #제5단# #제6단# #제7단# #제8단# #제9단#

```
2X 1= 2 3X 1= 3 4X 1= 4 5X 1= 5 6X 1= 6 7X 1= 7 8X 1= 8 9X 1= 9
2X 2= 4 3X 2= 6 4X 2= 8 5X 2=10 6X 2=12 7X 2=14 8X 2=16 9X 2=18
2X 3= 6 3X 3= 9 4X 3=12 5X 3=15 6X 3=18 7X 3=21 8X 3=24 9X 3=27
2X 4= 8 3X 4=12 4X 4=16 5X 4=20 6X 4=24 7X 4=28 8X 4=32 9X 4=36
2X 5=10 3X 5=15 4X 5=20 5X 5=25 6X 5=30 7X 5=35 8X 5=40 9X 5=45
2X 6=12 3X 6=18 4X 6=24 5X 6=30 6X 6=36 7X 6=42 8X 6=48 9X 6=54
2X 7=14 3X 7=21 4X 7=28 5X 7=35 6X 7=42 7X 7=49 8X 7=56 9X 7=63
2X 8=16 3X 8=24 4X 8=32 5X 8=40 6X 8=48 7X 8=56 8X 8=64 9X 8=72
2X 9=18 3X 9=27 4X 9=36 5X 9=45 6X 9=54 7X 9=63 8X 9=72 9X 9=81
```

[예제모음 30] 구구단을 파일에 출력

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05     FILE *wfp;           —— 파일 포인터와 변수를 선언한다.
06     int i, k;
07
08     wfp=fopen("c:\\temp\\gugu.txt", "w"); —— 쓰기 모드로 파일을 연다.
09     for( i = 2; i <= 9; i++ )           —— 첫 줄에 단 제목을 출력한다.
10         fprintf(wfp, " #제%d단# ", i);
11     fprintf(wfp, "\n\n");               —— 줄 넘김을 출력한다.
12     for( i = 1; i <= 9; i ++ )           —— 반복문을 돌면서 출력되는 구구단을
13     {                                     'gugu.txt' 파일에 저장한다.
14         for( k = 2; k <= 9; k ++ )
15         {
16             fprintf(wfp, "%2dX%2d=%2d ", k, i, k*i);
17         }
18         fprintf(wfp, "\n");
19     }
20
21     fclose(wfp);
22 }
```


[예제모음 31] 파일에서 읽어온 문자열을 파일에 반대 순서로 출력

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 를 누르고 저장한다).

실행 결과

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

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

[예제모음 31]파일에서 읽어온 문자열을 파일에 반대 순서로 출력

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 #include <string.h>
04 void main( )
05 {
06     FILE *rfp, *wfp;           — 파일 포인터를 선언한다.
07     char str1[200], str2[200]; — 입력 문자열, 출력 문자열, 변수를 선언한다.
08     int size, i;
09
10     rfp=fopen("c:\\temp\\in.txt", "r"); — 입력 파일과 출력 파일을 연다.
11     wfp=fopen("c:\\temp\\out.txt", "w");
12
13     while(1)                   — 무한 루프이다.
14     {
15         fgets(str1, 200, rfp); — 입력 파일의 문자열을 읽는다.
16
17         if(feof(rfp))          — 입력 파일의 끝이면 종료한다.
18             break;
19
20         size = strlen(str1);
```

[예제모음 31]파일에서 읽어온 문자열을 파일에 반대 순서로 출력

```
21     for(i=size-1; i >= 0; i--)
22         str2[size-1-i] = str1[i-1];
23
24     str2[size-1] = '\0';
25     fputs(str2, wfp);
26     fputs("\n", wfp);
27 }
28
29 fclose(rfp);
30 fclose(wfp);
31 }
```

—— '문자열 길이 -1'만큼 반복하며 입력 문자열과 출력 문자열의 위치를 바꾼다.

—— 출력 문자열의 맨 끝에 널 문자를 추가한다.

—— 출력 문자열을 출력 파일에 쓰고 줄 바꿈을 한다.

감사합니다!

