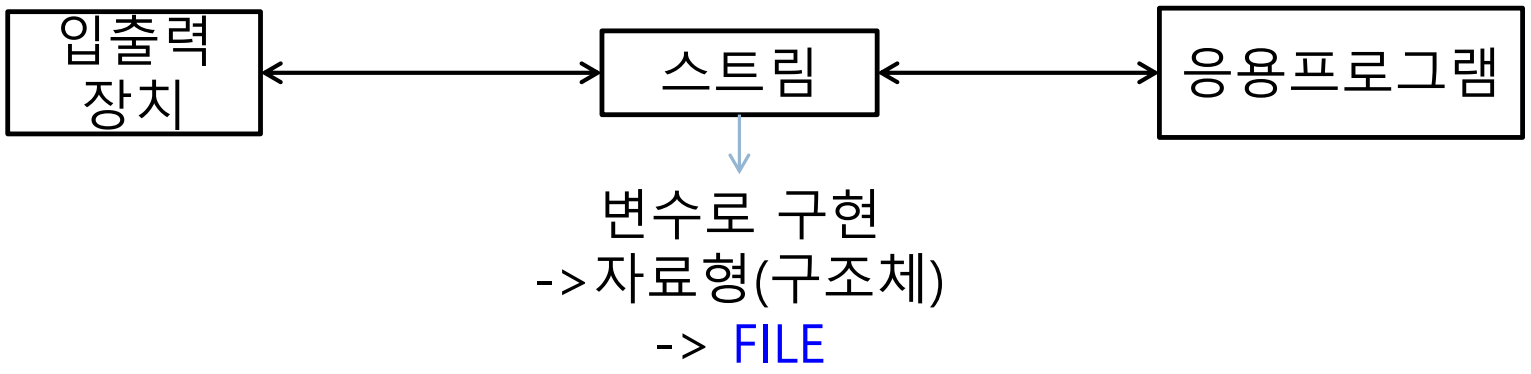
21장 문자와 문자열 관련함수

차례

스트림의 자료형 FILE

- 스트림은 소프트웨어적으로 하나의 변수(데이터)처럼 취급 -> 입출력 스트림을 나타내는 자료형이 필요 -> FILE(구조체)
- stdin, stdout, stderr -> 표준 스트림의 주소로 정의 -> 자료형이 FILE*
- 문자, 문자열 전용 입출력 함수에서 주소를 이용하여 스트림에 접근함



문자 출력 전용함수: putchar, fputc

- 하나의 문자만 출력하는 전용 함수-> printf 함수보다 처리속도 빠름
- fputc의 두 번째 인자로 stdout이 전달되면 putchar 함수와 동일

```
#include <stdio.h>
int putchar(int c);
int fputc(int c, FILE * stream);
```

➜ 함수호출 성공 시 쓰여진 문자정보가, 실패 시 EOF 반환

문자 출력 전용함수 : putchar

- int putchar(int c);
  - 매개변수 c에 전달된 문자를 모니터(표준 출력 스트림)에 출력
  - 정상동작시 화면에 출력한 문자를 반환함, 오류 발생시 EOF(-1)를 반환함
  - 출력 버퍼를 사용함, stdio.h 헤더파일을 포함해야 함

```
#include <stdio.h>
int ch = 'A';                  // 정수형에 주의
putchar(ch);                   // 인자에 문자를 저장한 변수를 전달
putchar('B');                  // 인자에 문자상수를 직접 전달
                               // printf("%c", ch); 와 같음
```

문자 출력 전용함수 : fputc

- int fputc(int c, FILE* stream);
  - 매개변수 c에 전달된 문자를 stream이 가리키는 출력 스트림에 출력
  - 정상동작시 화면에 출력한 문자를 반환함, 오류 발생시 EOF(-1)를 반환함
  - 출력 버퍼를 사용함, stdio.h 헤더파일을 포함해야 함
  - putchar(c); => fputc(c, stdout);

```
#include <stdio.h>
int ch = 'A';                  // 정수형에 주의
fputc(ch, stdout);             // 변수에 저장된 문자를 모니터에 출력
fputc('B', stdout);            // 문자상수 'B'를 모니터에 출력
```

문자 입력 전용함수 : getchar, fgetc

- 하나의 문자만 입력 받는 전용 함수-> scanf 함수보다 처리속도 빠름
- getchar 함수와 fgetc 함수의 관계는 putchar 함수와 fputc 함수의 관계와 같음

```
#include <stdio.h>
int getchar(void);
int fgetc(FILE * stream);
```

➡ 파일의 끝에 도달하거나 함수호출 실패 시 EOF 반환

문자 입력 전용함수 : getchar

- int getchar(void);
  - 키보드(표준 입력 스트림)에서 입력된 문자의 아스키코드를 반환
  - 읽기 오류 또는 파일 끝(Ctrl-z)이 입력되면 EOF(-1)를 반환
  - 입력 버퍼를 사용함, stdio.h 헤더파일을 포함해야 함

```
#include <stdio.h>
int ch;                    // 정수형에 주의 -> char 선언가능
ch = getchar();            // 키보드에서 입력된 문자를 변수를 저장
                           // scanf("%c", &ch); 와 같음
```

문자 입력 전용함수 : fgetc

- int fgetc(FILE* stream);
  - stream이 가리키는 입력 스트림에서 하나의 문자를 읽어서 반환
  - 읽기 오류 또는 파일 끝(Ctrl-z)이 입력되면 EOF(-1)를 반환
  - 출력 버퍼를 사용함, stdio.h 헤더파일을 포함해야 함
  - ch = getchar(); => ch = fgetc(stdin);

```
#include <stdio.h>
int ch;                     // 정수형에 주의
ch = fgetc(stdin);          // stdin에서 한 문자를 읽어서 ch에 저장
```

예제1 : 문자 입출력

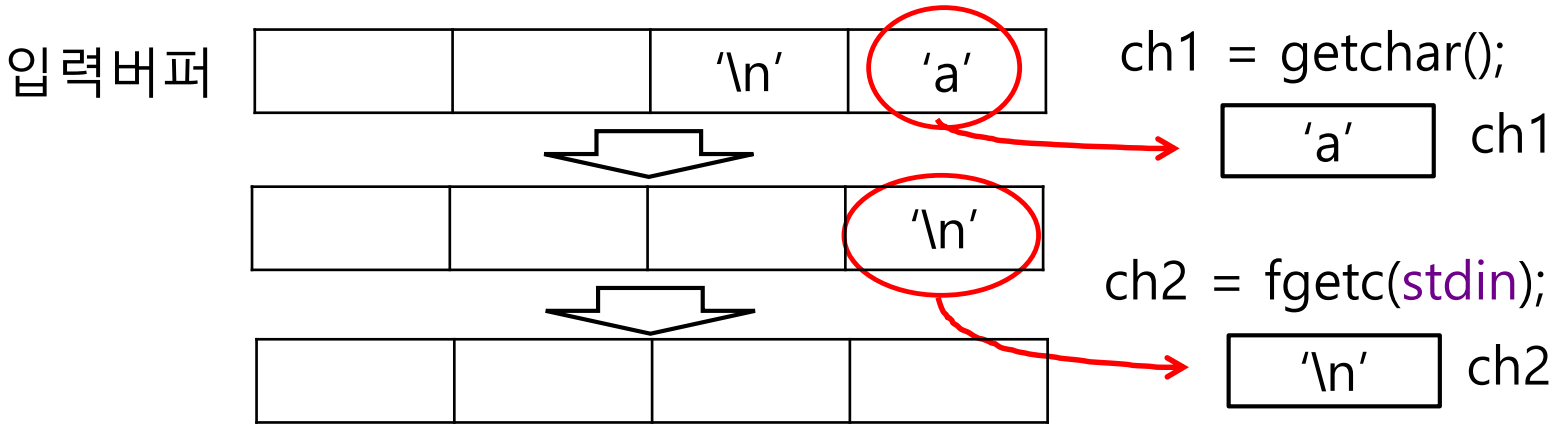

```c
#include <stdio.h>
int main(void)
{
	int ch1, ch2;
	ch1 = getchar();		// 문자 입력
	ch2 = fgetc(stdin);		// 엔터 키 입력

	putchar(ch1);			// 문자 출력
	fputc(ch2, stdout);		// 엔터 키 출력
	return 0;
}
```

```
a<엔터>
a
```

- *문자의 입력을 완성하는 엔터키의 입력도 하나의 문자로 인식-> 따라서 이 역시도 입출력이 가능*
- *문자를 int형 변수에 저장하는 이유는 EOF를 설명하면서 함께 설명*

예제 1

- ch1 = getchar(); -> 'a'를 변수 ch1에 저장
- ch2 = fgetc(stdin); -> 엔터키('₩n')를 변수 ch2에 저장



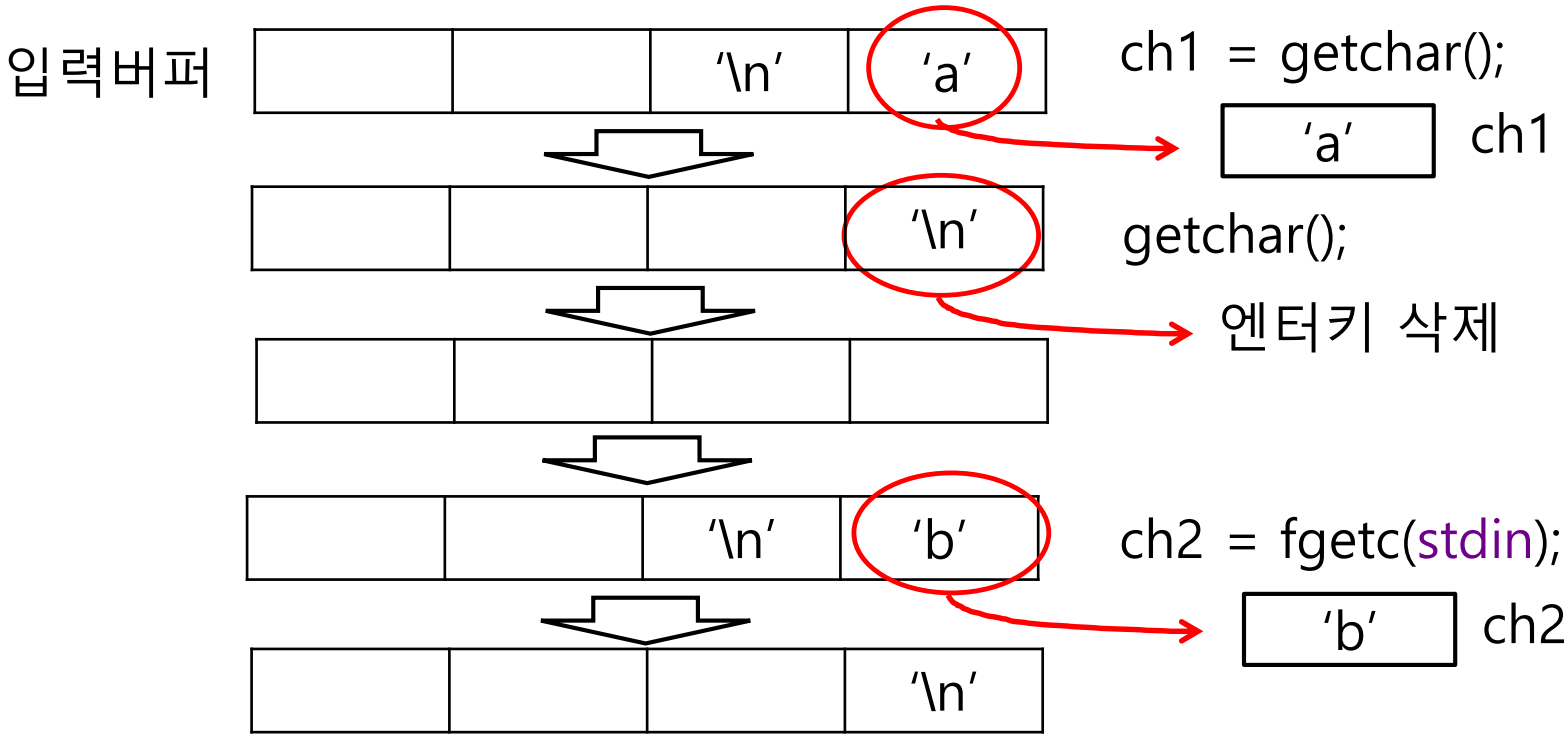
예제2 : 문자 입출력


```c
#include <stdio.h>
int main(void)
{
	int ch1, ch2;

	ch1 = getchar();		// 문자 입력
	getchar();			// 엔터키 삭제
	ch2 = fgetc(stdin);		// 문자 입력

	putchar(ch1);			// 문자 출력
	putchar('₩n');			// 줄바꿈
	fputc(ch2, stdout);		// 문자 출력
	return 0;
}
```

```
a <엔터>
b <엔터>
a
b
```

예제2

- ch1 = getchar(); -> 'a'를 변수 ch1에 저장
- ch2 = fgetc(stdin); -> 엔터키('₩n')를 변수 ch2에 저장



문자 입출력에서의 EOF

- EOF의 의미
  - EOF는 End Of File의 약자로서, 파일의 끝을 표현하기 위해서 정의해 놓은 상수.
  - 파일을 대상으로 fgetc 함수가 호출되었을 때 파일에 끝에 도달을 하면 EOF가 반환됨
- fgetc, getchar 함수호출시 EOF를 반환하는 경우
  - 장치의 고장, 운영체제 오류 등으로 인한 함수의 동작오류
  - Windows에서 Ctrl+z 키, Linux에서 Ctrl+d 키가 입력이 되는 경우
  - Ctrl+z 를 입력하는 경우 엔터키를 입력할 필요 없음

예제3: 문자 입출력에서의 EOF

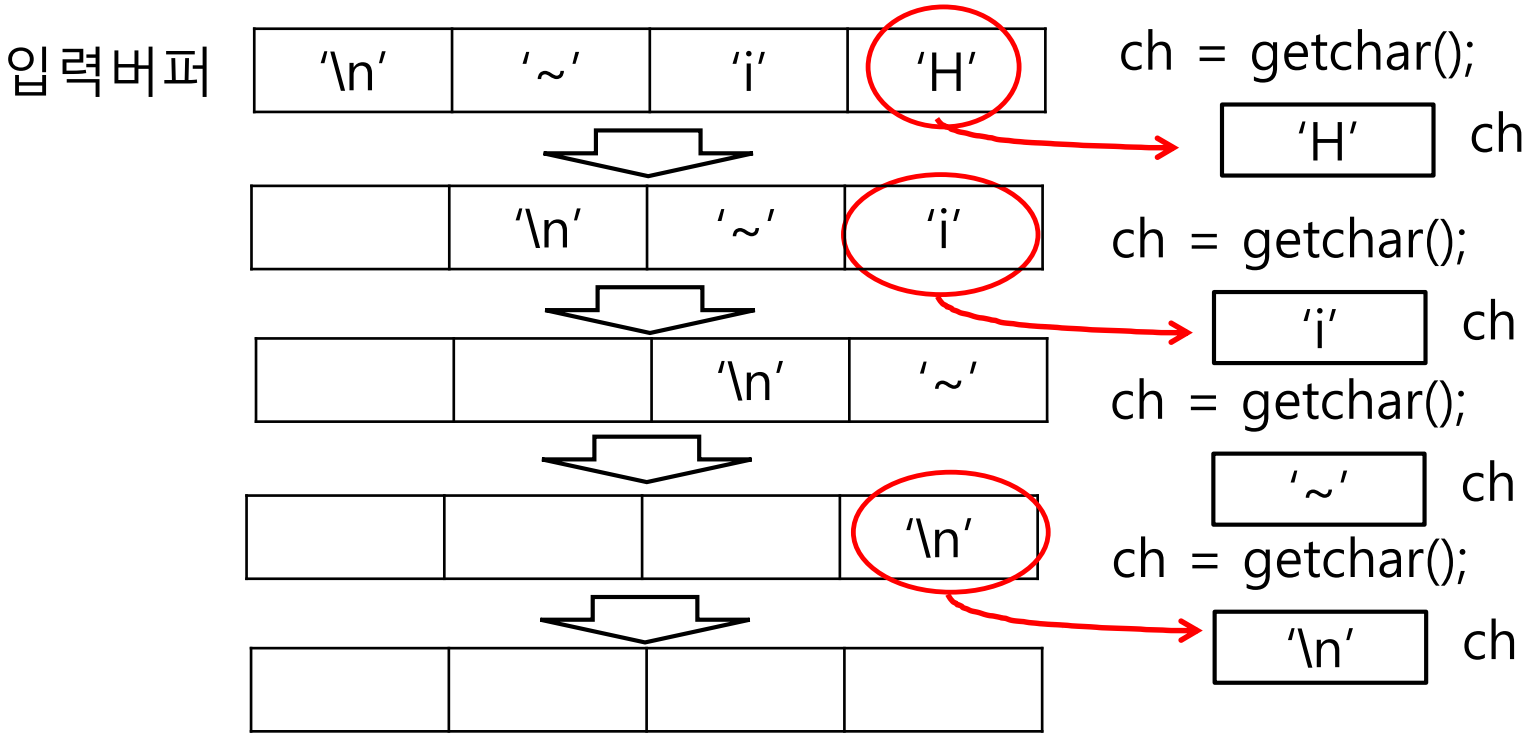

```
#include <stdio.h>
int main(void)
{
	int ch;

	while (1)
	{
		ch = getchar();
		if (ch == EOF) break;
		putchar(ch);
	}
	return 0;
}
```

```
Hi~<엔터>
Hi~
I like C lang.<엔터>
I like C lang.
^Z
```

- *키보드에는 EOF가 존재하지 않음, 따라서 EOF를 Ctrl+z 키 또는 Ctrl+d 키로 약속해 놓음*
- *예제에서 보이듯이 하나의 문장이 입력되어도 문장을 이루는 모든 문자들이 반복된 getchar 함수의 호출을 통해서 입력될 수 있음*

예제3

- 2~4번째 getchar()함수 호출은 사용자 입력없이 바로 리턴



반환형이 int이고, int형 변수에 문자를 담는 이유는?

- 반환형이 char형이 아닌 int형인 이유는?
  - char형은 예외적으로 signed char가 아닌 unsiged char로 표현하는 컴파일러가 존재한다.
  - 파일의 끝에 도달했을 때 반환하는 EOF는 -1로 정의되어 있다.
  - char를 unsigend char로 표현하는 컴파일러는 EOF에 해당하는 -1을 반환하지 못한다.
  - int는 모든 컴파일러가 signed int로 처리한다. 따라서 -1의 반환에 무리가 없다.

```
int getchar(void);
int fgetc(FILE * stream);
```

문자열 출력 전용함수: puts, fputs

- 문자열만 출력하는 전용함수 -> printf함수보다 처리속도가 빠름
- puts 함수가 호출되면 문자열 출력 후 자동으로 줄바꿈이 이뤄지지만, fputs 함수는 문자열 출력 후 자동으로 줄바꿈이 이뤄지지 않음

```
#include <stdio.h>
int puts(const char * s);
int fputs(const char * s, FILE * stream);
```

➜ 성공 시 0이 아닌 값을, 실패 시 EOF 반환

문자열 출력 전용함수: puts

- int puts(const char* s);
  - 매개변수 s가 가리키는 문자열을 모니터(표준 출력 스트림)에 출력
  - 널문자('₩0')는 줄바꿈 문자('₩n')로 변경하여 출력-> 줄바꿈 자동 발생
  - 성공시 음수가 아닌 값을 반환, 오류 발생시 EOF(-1)를 반환
  - 버퍼이용, stdio.h 헤더파일을 포함해야 함

```
#include <stdio.h>
char str[10] = "hello";
puts(str);                  // 인자에 문자열의 주소 전달
puts("hello");              // 인자에 문자열상수를 직접 전달
                            // printf("%s", str); 와 같음
```

문자열 출력 전용함수: fputs

- int fputs(const char* s, FILE* stream);
  - 매개변수 s가 가리키는 문자열을 stream이 가리키는 출력 스트림에 출력
  - 자동 줄바꿈 안함
  - 성공시 음수가 아닌 값을 반환, 오류 발생시 EOF(-1)를 반환
  - 버퍼이용, stdio.h 헤더파일을 포함해야 함

```
#include <stdio.h>
char str[10] = "hello";
fputs(str, stdout);          // 문자열 str을 stdout에 출력
fputs("hello", stdout);      // 문자열상수를 직접 전달
```

예제4: puts, fputs


```c
#include <stdio.h>
int main(void)
{
	char str[] = "Simple String";

	printf("1. puts test ------ ₩n");
	puts(str);                          // 자동 줄바꿈
	puts("So Simple String");           // 자동 줄바꿈

	printf("2. fputs test ----- ₩n");
	fputs(str, stdout); printf("₩n");                          // 자동 줄바꿈 안해줌
	fputs("So Simple String", stdout); printf("₩n");           // 자동 줄바꿈 안해줌

	printf("3. end of main ----₩n");
	return 0;
}
```

```
1. puts test ------
Simple String
So Simple String
2. fputs test -----
Simple String
So Simple String
3. end of main ----
```

문자열 입력 전용함수: gets, fgets

- 문자열만 입력해주는 전용함수 -> scanf함수보다 처리속도 빠름
- gets함수는 입력되는 문자열의 길이가 배열길이보다 클 경우 할당 받지 않은 메모리를 참조하는 오류가 발생
- fgets함수는 stdin으로부터 문자열을 입력 받아서 s에 저장을 하되, 널 문자를 포함하여 지정된 크기만큼만 저장해줌

```
#include <stdio.h>
char * gets(char * s);
char * fgets(char * s, int n, FILE * stream);
```

➜ 파일의 끝에 도달하거나 함수호출 실패 시 NULL 포인터 반환

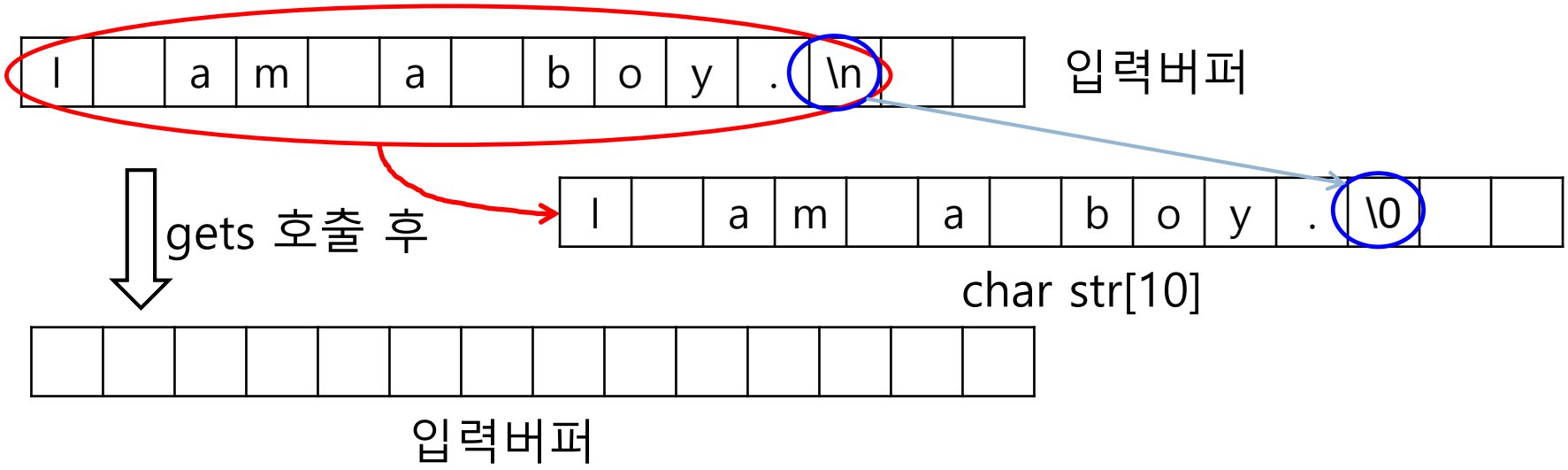
문자열 입력 전용함수: gets

- char* gets(const char* str);
  - 키보드(표준 입력 스트림)에서 입력 받은 문자열을 주소 str에 저장
  - 엔터키가 입력되기전까지 **공백을 포함한 문자열을 저장**
  - **엔터키('₩n')는 읽어서 저장하지 않고 대신에 널문자('₩0')를 저장함**
  - **엔터키('₩n')를 입력 버퍼에 남기지 않음**
  - 정상동작시 인자를 그대로 리턴, 오류시 NULL 포인터((void*)0)를 리턴
  - 입력 버퍼를 사용함, stdio.h 헤더파일을 포함해야 함

```
#include <stdio.h>
char str[20];
gets(str);                    // 인자에 배열주소 전달
```

문자열 입력 전용함수: gets

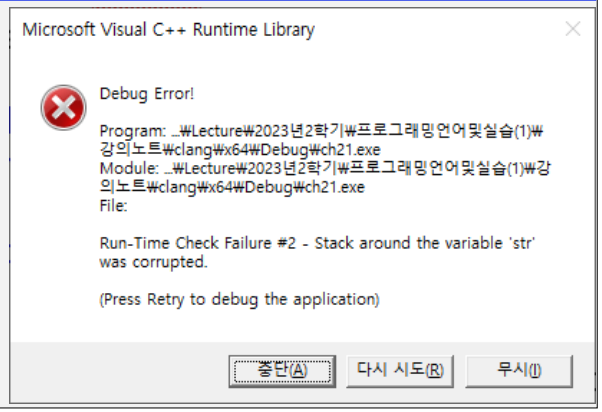
- gets 함수는 '₩n'도 읽고 널문자로 변환하여 배열에 저장 -> **버퍼에는 '₩n'는 남지 않음**



예제5: gets 함수 오류


```c
#include <stdio.h>
int main(void)
{
	char str[12];
	int i;

	for (i = 0; i < 3; i++)
	{
		gets(str);
		printf("Read %d: %s ₩n", i + 1, str);
	}
	return 0;
}
```

```
hello world<엔터>
Read 1: hello world
I am a boy.<엔터>
Read 2: I am a boy.
I am a boy, you are a girl.<엔터>
Read 3: I am a boy, you are a girl.
```



- *공백을 포함한 문자열 입력가능*
- *배열의 크기를 초과하는 문자열 입력시 실행 오류 발생함(입력문자열길이+1<=배열크기)*
- *할당된 공간을 초과하는 것을 체크해주지 않음*

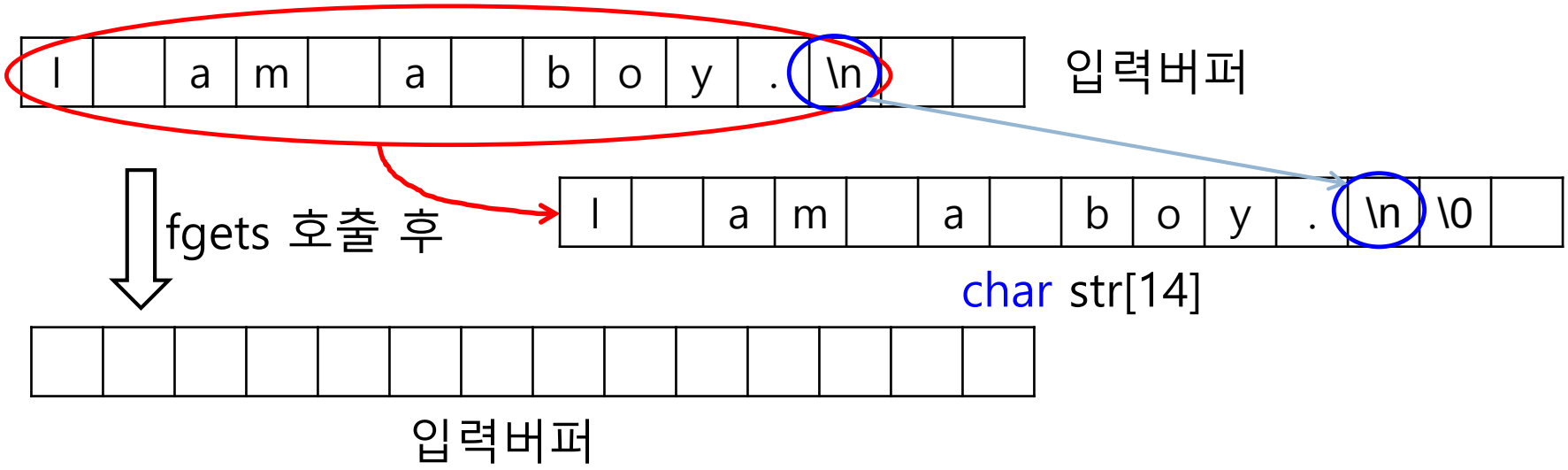
문자열 입력 전용함수: fgets

- char* fgets(const char* str, int n, FILE* stream);
  - stream이 가리키는 입력 스트림에서 문자열을 읽어서 주소 str에 저장, 공백포함한 문자열을 저장가능
  - **(n-1)개의 문자를 읽을 때까지 또는 첫번째 줄바꿈 문자('\n')를 만날 때까지 또는 스트림의 끝을 만날 때까지 문자열을 읽어 저장하고 줄바꿈 문자를 만나면 문자열에 저장함, 마지막에 널문자('\0')를 저장함**
  - 정상동작시 str를 리턴, 파일 끝을 만나거나 오류시 NULL((void*)0)을 리턴
  - 입력 버퍼를 사용함, stdio.h 헤더파일을 포함해야 함

```
#include <stdio.h>
char str[20];
fgets(str, sizeof(str), stdin);        // 인자에 주소와 문자열크기 전달
```

문자열 입력 전용함수: fgets

- fgets 함수는 '₩n'도 읽어서 배열에 저장하고 마지막에 널문자 저장 -> 버퍼에는 '₩n'는 남지 않음



예제6:fgets 함수


```c
#include <stdio.h>
int main(void)
{
	char str[7];
	int i;

	for (i = 0; i < 3; i++)
	{
		fgets(str, sizeof(str), stdin);
		printf("Read %d: %s ₩n", i + 1, str);
	}
	return 0;
}
```

```
12345678901234567890 <엔터>
Read 1: 123456
Read 2: 789012
Read 3: 345678
```

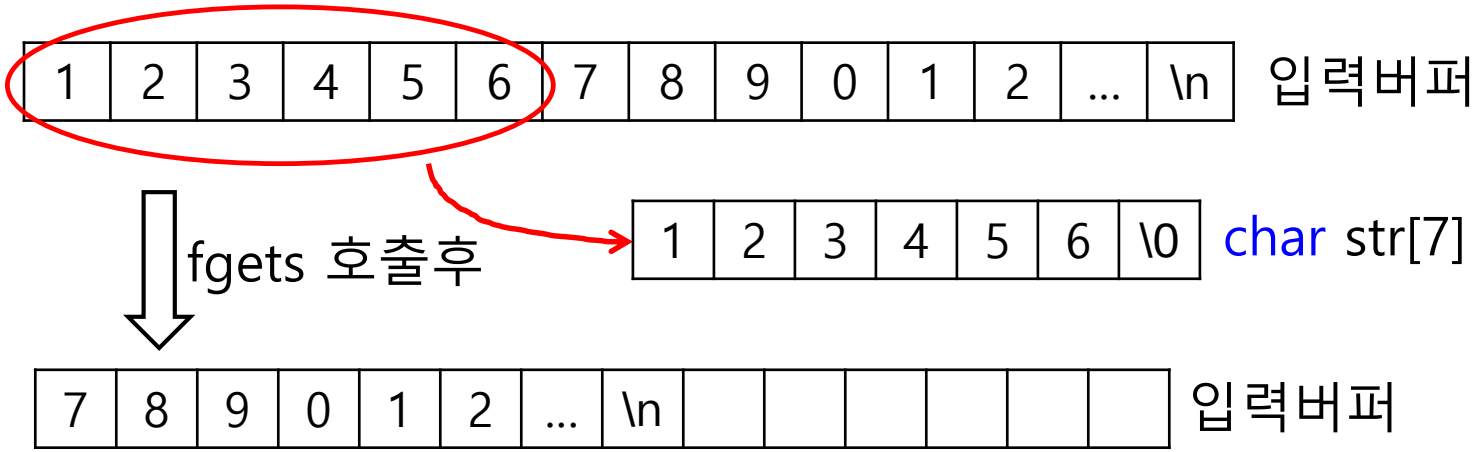
```
Y & I <엔터>
Read 1: Y & I

ha ha <엔터>
Read 2: ha ha

^^ -- <엔터>
Read 3: ^^ --
```

```
We <엔터>
Read 1: We

like <엔터>
Read 2: like

you <엔터>
Read 3: you
```

- *첫번째결과 : 문자열의 길이가 6보다 큰 경우는 잘라서 저장*
- *2,3번째 결과 : 엔터키 포함한 입력문자열이 7보다 작은 경우는 엔터키 까지만 저장->2번 줄바꿈 발생*

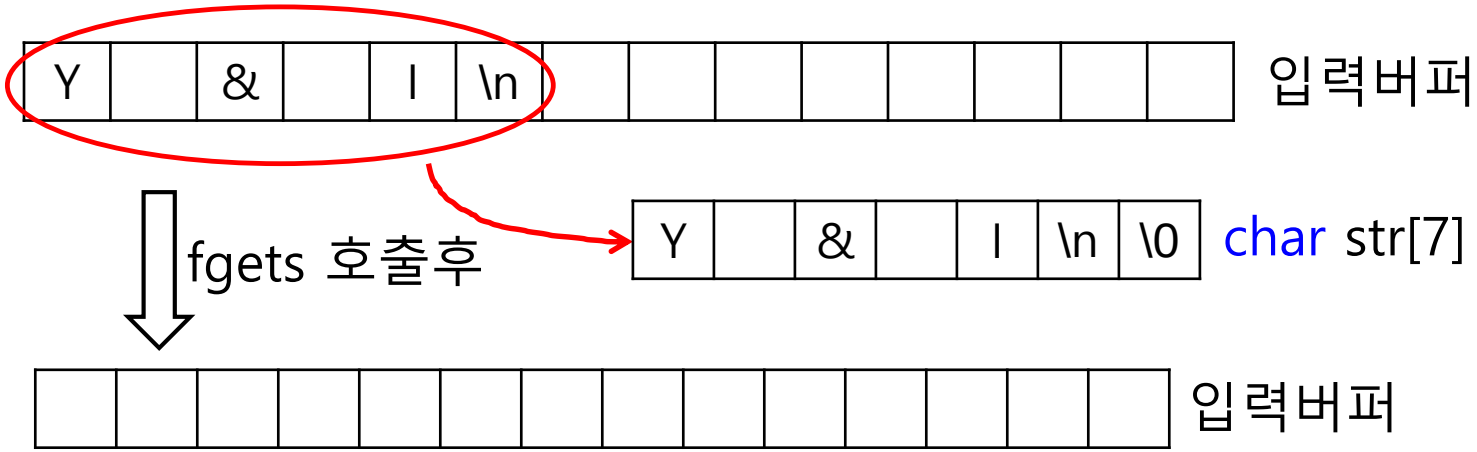
예제6:fgets 함수

- 첫번째 결과 : 6문자만 입력 받음, 엔터키 포함 안됨



예제6:fgets 함수

- 두번째 결과 : 6문자만 입력 받음, 엔터키 포함됨



예제7(선택): 버퍼에 남은 엔터키로 인한 오류


```c
#include <stdio.h>
int main(void)
{
	char perID[7];
	char name[10];

	fputs("주민번호 앞 6자리 입력: ", stdout);
	fgets(perID, sizeof(perID), stdin);

	fputs("이름 입력: ", stdout);
	fgets(name, sizeof(name), stdin);

	printf("주민번호: %s₩n", perID);
	printf("이름: %s₩n", name);
	return 0;
}
```

```
주민번호 앞 6자리 입력: 950915<엔터>
이름 입력: 주민번호: 950915
이름:
```

```
주민번호 앞 6자리 입력: 950915-1122345<엔터>
이름 입력: 주민번호: 950915
이름: -1122345
```

예제8(선택): 입력버퍼 비우기


```c
#include <stdio.h>
void ClearLineFromReadBuffer(void)
{
	while (getchar() != '₩n');
}
int main(void)
{
	char perID[7];
	char name[10];

	fputs("주민번호 앞 6자리 입력: ", stdout);
	fgets(perID, sizeof(perID), stdin);
	ClearLineFromReadBuffer();	// 입력버퍼 비우기

	fputs("이름 입력: ", stdout);
	fgets(name, sizeof(name), stdin);

	printf("주민번호: %s₩n", perID);
	printf("이름: %s₩n", name);
	return 0;
}
```

```
주민번호 앞 6자리 입력: 950915 <엔터>
이름 입력: 홍길동
주민번호: 950915
이름: 홍길동
```

```
주민번호 앞 6자리 입력: 950915-1122345 <엔터>
이름 입력: 홍길동
주민번호: 950915
이름: 홍길동
```

함수별 종료문자 '\n' 처리방법

- scanf("%d",...)
  - 호출전에 버퍼에 남아 있는 '₩n' 값을 무시한다.
  - 버퍼에서 읽은 값은 지워주고 종료 문자로 입력한 '₩n' 는 버퍼에 남김
- getchar()
  - 호출전에 버퍼에 남아 있는 '₩n' 값도 문자로 인식하여 입력 받음
  - 버퍼에서 읽은 값은 지워주고 종료 문자로 입력한 '₩n' 는 버퍼에 남김
- scanf("%c",...)
  - 호출전에 버퍼에 남아 있는 '₩n' 값도 문자로 인식하여 입력 받음.
  - 버퍼에서 읽은 값은 지워주고 종료 문자로 입력한 '₩n' 는 버퍼에 남김
- scanf("%s",...)
  - 호출전에 버퍼에 남아있는 '₩n' 값을 무시한다.
  - 버퍼에서 읽은 값은 지워주고 종료 문자로 입력한 '₩n' 는 버퍼에 남김
- gets(), fgets()
  - 호출전에 버퍼에 남아있는 '₩n' 값을 종료 문자로 인식한다.
  - 버퍼에서 읽은 값은 지워주고 종료 문자로 입력한 '₩n' 도 버퍼에서 지워줌

실습과제 1

- 공백을 포함한 문자열을 입력 받을 때 사용가능한 함수는?
- gets 함수사용시 가장 큰 문제점은 무엇인가?
- fgets 함수 사용시 문제점은 무엇인가?
- 널문자('₩0')와 널포인터(NULL)의 용도를 설명하시오.

실습과제2

- 키보드로부터 문자를 입력 받아 화면에 출력해주는 프로그램을 작성하시오. ‘q’ 또는 ‘Q’를 입력하면 종료한다.
- 반드시 문자 입출력 전용함수만 사용하라.
- 힌트: 무한 루프(while(1){...})와 if(종료조건) break문을 이용하라.

```
문자입력 : a<엔터>
입력된 문자는 a
문자입력: 1<엔터>
입력된 문자는 1
문자입력: C<엔터>
입력된 문자는 C
문자입력: q<엔터>
```

실습과제3

- 아래 결과처럼 실행되도록 문자열 입출력 전용함수를 이용하여 코드를 작성하시오. 한글은 한문자를 2바이트로 저장함

이름: 홍길동<엔터>
주소: 군산시 대학로 558<엔터>
자동차번호: 52어6746<엔터>
<결과출력>
이름:홍길동
주소:군산시 대학로 558
자동차번호: 52어6746

실습과제 4

- 아래처럼 공백을 포함하는 문자열을 입력 받아 대소문자를 서로 변환하여 출력하시오.
- 문자, 문자열 입출력 전용함수만 사용할 것
- 힌트: 널('₩0')문자를 만날 때까지 반복하면서 소문자를 대문자로 변환 하시오. 대문자 = 소문자 – ('a'-'A'), 영문자가 아니면 그대로 출력

문자열입력 : Hello World<엔터>
변환결과 : hELLO wORLD

실습과제 5

- 아래 코드에서 fgets함수는 엔터키까지 저장해주므로 출력하면 줄 바꿈이 발생한다. 엔터키를 널문자로 변경하는 코드를 추가하여 줄 바꿈이 발생하지 않도록 해보시오.

```
#include <stdio.h>
int main(void)
{
	char str[20];
	printf("문자열입력:");
	fgets(str, sizeof(str), stdin);
	// 배열에서 '₩n'을 널문자('₩0')로 변환
	printf("%s", str);
	return 0;
}
```

문자열입력: hello <*엔터*>
hello

문자열입력: hello <*엔터*>
hello

과제제출방법

- 소스코드, 라인단위의 주석, 실행결과를 포함하는 pdf파일을 작성한 후 eclass 과제 게시판에 업로드, **반드시 하나의 pdf파일로 업로드할 것**
- 기한 : 과제 게시판에 마감시간 참조
- 실행결과를 캡쳐할 때 글자를 알아보기 쉽게 확대해서 캡쳐할 것.
- 소스코드의 첫 부분은 아래처럼 제목,날짜,작성자(학번,이름)를 작성할 것

```
// ************************************************
//    제   목 :  정수 4개의 평균을 구하는 프로그램
//    날   짜 :  2023년 9월10일
//    작성자 :  15010101 홍길동
// ************************************************

// 소스코드 작성
```