

컴퓨터 프로그래밍2

문자 입출력

장서윤 pineai@cnu.ac.kr

문자 입출력

- 모든 문자는 숫자로 저장되고 연산 가능
- `%c`는 화이트 스페이스인 공백문자, 탭문자, 개행문자도 입력
- `%c` 앞에 공백 사용하면 화이트 스페이스는 입력에서 제외 가능
- `getchar`, `putchar`는 문자 전용 입출력 함수
- 아스키 코드값은 문자를 1대 1로 대응시킨 숫자

문자 입출력

▶아스키 코드값은 문자를 1대 1로 대응시킨 숫자

표 11-1 문자 입출력 함수

구분	사용 예	기능
입력	<code>char ch;</code> <code>scanf("%c", &ch);</code>	char형 변수 사용 %c 변환문자로 입력 공백문자, 탭문자, 개행문자도 입력
	<code>int ch;</code> <code>ch = getchar();</code>	int형 변수 사용 입력 문자의 아스키 코드값 반환 공백문자, 탭문자, 개행문자도 입력
출력	<code>printf("%c", ch);</code> <code>putchar(ch);</code>	%c 변환문자 사용 문자 출력 전용함수, 출력할 문자 전달

문자 상수 구현 방법

- ▶ 프로그램에서 문자 사용할 때는 항상 양쪽에 작은 따옴표
 - ▶ 컴파일러는 a를 변수명으로 해석, 'a'는 문자 상수로 해석
 - ▶ 컴파일 후에는 더 이상 의미 없음
- ▶ 문자는 컴파일 과정에서 약속된 정수값인 아스키 코드(ASCII code)값으로 바뀜
 - ▶ ex) 문자 상수 'a' -> 정수값 97
- ▶ 문자는 메모리에 저장되는 방식이 정수와 같음
 - ▶ int형 변수에 저장하고 정수처럼 연산 가능

문자 상수 구현 방법

예제 11-1 정수처럼 사용되는 문자

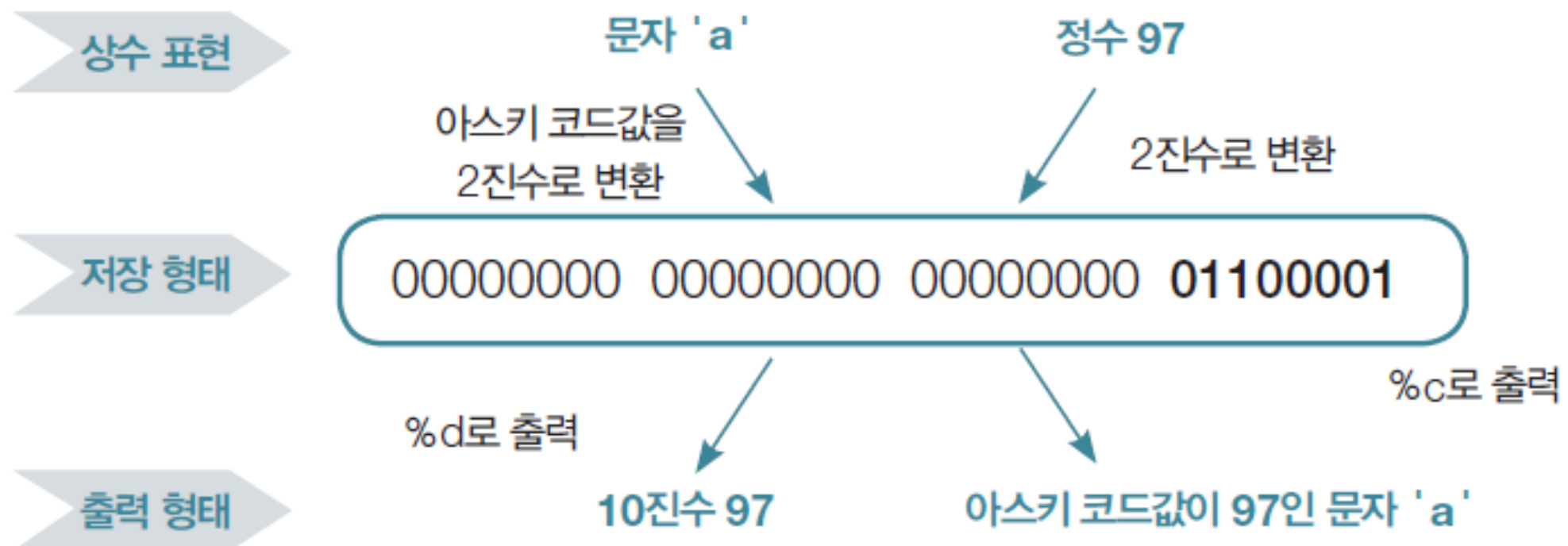
```
1. #include <stdio.h>
2.
3. int main(void)
4. {
5.     int ch;                // int형 변수
6.
7.     printf("문자 a의 아스키 코드값 : %d\n", 'a');
8.     printf("아스키 코드값이 97인 문자 : %c\n", 97);
9.     printf("문자 상수의 크기 : %d바이트\n", sizeof('a'));
10.
11.    ch = 'a';               // 문자를 int형 변수에 대입
12.    ch++;                   // 아스키 코드값 1 증가
13.    printf("문자 %c의 아스키 코드값 : %d\n", ch, ch);
14.    printf("%c" , ch); // b 출력
15.    return 0;
16. }
```

실행 결과

- 문자 a의 아스키 코드값 : 97
- 아스키 코드값이 97인 문자 : a
- 문자 상수의 크기 : 4바이트
- 문자 b의 아스키 코드값 : 98

문자 상수 구현 방법

- ▶ 문자가 정수와 같은 형태의 데이터라 할지라도 항상 작은 따옴표 써서 문자 상수로 읽기 쉽게 작성할 것



문자 상수 구현 방법

- ▶ 증가 연산 통해 다음 아스키 코드값 갖는 문자 확인 가능
- ▶ 문자 'a'값 증가시키며 반복 출력하면 모든 알파벳 출력하는 것도 가능

```
ch = 'a';           // ch에 첫 문자로 초기화
while(ch <= 'z')    // 아스키 코드값이 마지막 문자보다 작거나 같은 동안
{
    printf("%c", ch); // 문자 출력
    ch++;             // 아스키 코드값 1 증가
}
```

아스키 코드

▶아스키 코드란?

▶128개의 문자를 0~127의 숫자 중에 각각 어떤 값으로 표현할지 정의한 것

▶ 전체 아스키 코드값은 부록 참고

▶아스키 코드 요약

표 11-2 아스키 코드표 요약

종류	문자 상수	아스키 코드값	출력할 때
숫자 문자 (10개)	'0' ~ '9'	48 ~ 57	문자 출력
대문자 (26개)	'A' ~ 'Z'	65 ~ 90	문자 출력
소문자 (26개)	'a' ~ 'z'	97 ~ 122	문자 출력
특수 문자 (33개)	' ' (공백), '\$', '&' ...	32, 36, 38 ...	문자 출력
제어 문자 (33개)	'\0', '\t', '\n', '\r' ...	0, 9, 10, 13 ...	제어 기능 수행

아스키 코드

▶아스키 코드의 특징

- ▶알파벳과 숫자는 각각 연속된 아스키 코드값 가짐
- ▶소문자가 대문자보다 아스키 코드값이 큼
- ▶제어 문자는 백슬래시와 함께 표시하며 출력할 때 그 기능 수행

아스키 코드

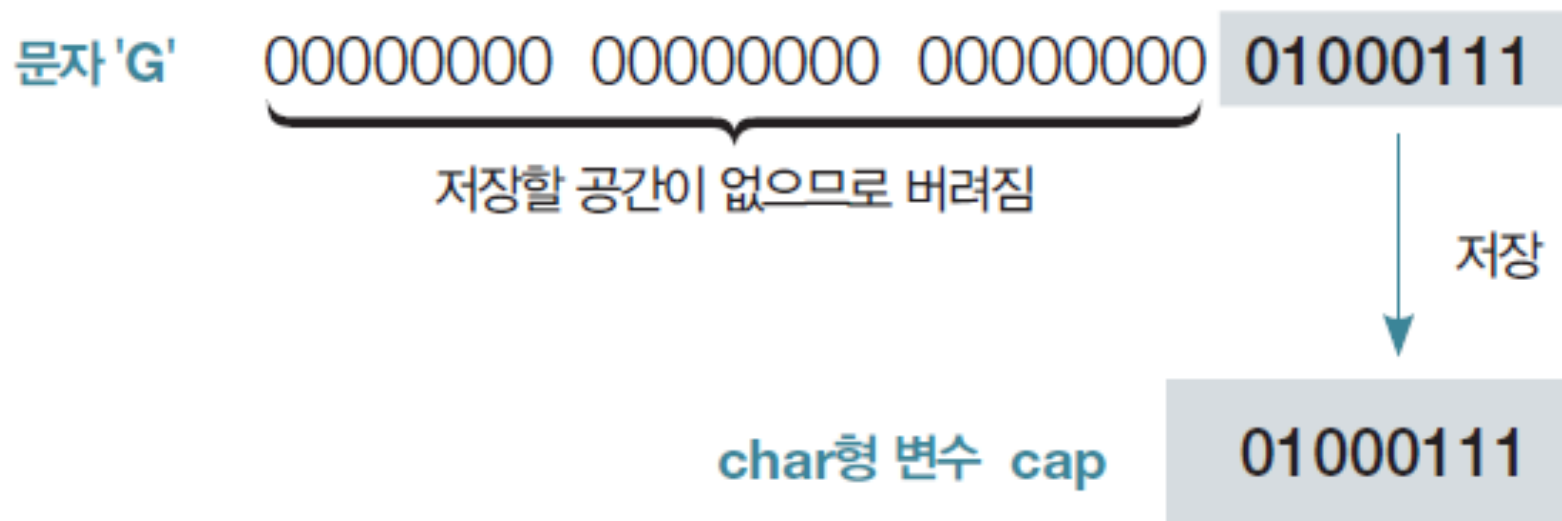
.....

예제 11-2 대문자를 소문자로 바꾸는 프로그램

```
1. #include <stdio.h>
2.
3. int main(void)
4. {
5.     char small, cap = 'G';           // char형 변수 선언과 초기화
6.
7.     if( (cap >= 'A') && (cap <= 'Z') ) // 대문자 범위라면
8.     {
9.         small = cap + ('a' - 'A');    // 대,소문자의 차이를 더해 소문자로 변환
10.    }
11.    printf("대문자 : %c %c", cap, '\n'); // '\n'를 %c로 출력하면 줄이 바뀐다.
12.    printf("소문자 : %c", small);
13.
14.    return 0;
15. }
```

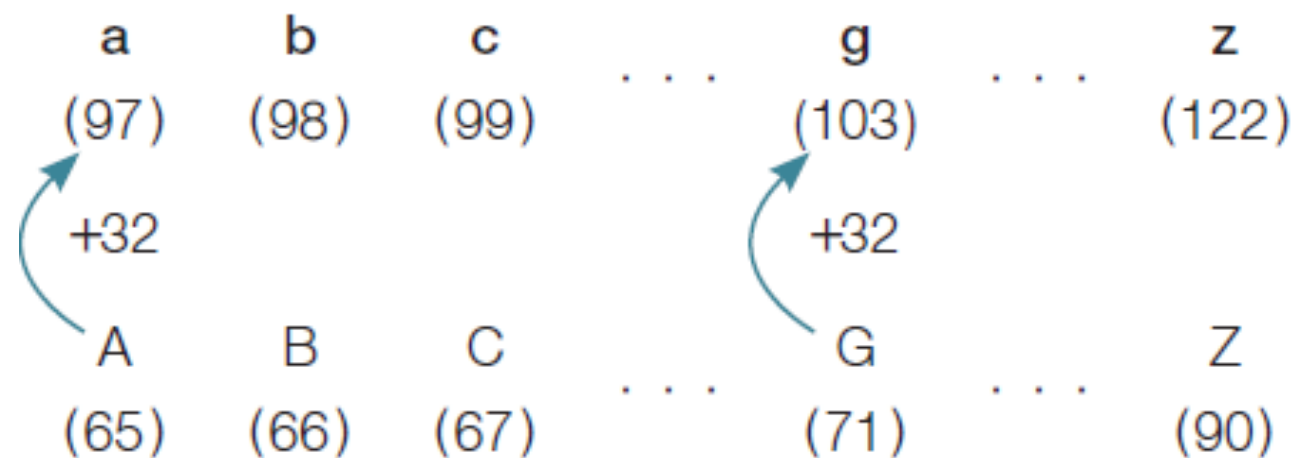
실행
결과
대문자 : G
소문자 : g

- 컴파일러는 문자에서 아스키 코드값을 갖는 오른쪽 1바이트만 변수에 저장하고 남은 바이트는 버림



아스키 코드

- ▶ cap에 저장된 대문자를 소문자로 바꾸어 변수 small에 저장
- ▶ 소문자에서 대문자를 뺀 차 활용
 - ▶ 'a' - 'A'의 값을 대문자 'G'에 더하면 소문자 'g' 구할 수 있음



아스키 코드

- ▶ 제어 문자를 프로그램에서 상수로 쓸 때
 - ▶ 백슬래시와 제어 기능을 암시하는 문자 함께 사용 제어
 - ▶ 문자는 형태가 없으므로 %c로 출력 시 해당 제어 기능 수행
 - ▶ ex) 줄 바꾸는 문자는 new line의 n 따서 ‘\n’

SCANF 함수를 사용한 문자 입력

- ▶ scanf 함수로 문자 입력할 때 - %c 변환문자열 사용
- ▶ %c
 - ▶ 알파벳이나 숫자 모양의 문자 등 형태가 있는 문자 입력
 - ▶ 공백문자 (space), 탭문자 (tab), 개행문자 (enter)도 입력
 - ▶ 숫자 입력할 때 값 구분하기 위해 사용
 - ▶ 문자 입력할 때는 그 자체가 하나의 입력 데이터

SCANF 함수를 사용한 문자 입력

예제 11-3 제어 문자의 입력

```
1. #include <stdio.h>
2.
3. int main(void)
4. {
5.     char ch1, ch2;
6.
7.     scanf("%c%c", &ch1, &ch2);    // 두 개의 문자 연속 입력
8.     printf("[%c%c]", ch1, ch2);    // 입력된 문자 출력
9.
10.    return 0;
11. }
```

실행
결과

[실.행.결.과 1 - a와 b를 연속으로 입력하고 엔터를 친다.]

ab

[ab]

[실.행.결.과 2 - a와 공백을 연속으로 입력하고 엔터를 친다.]

a

[a]

[실.행.결.과 3 - a만 입력하고 엔터를 친다.]

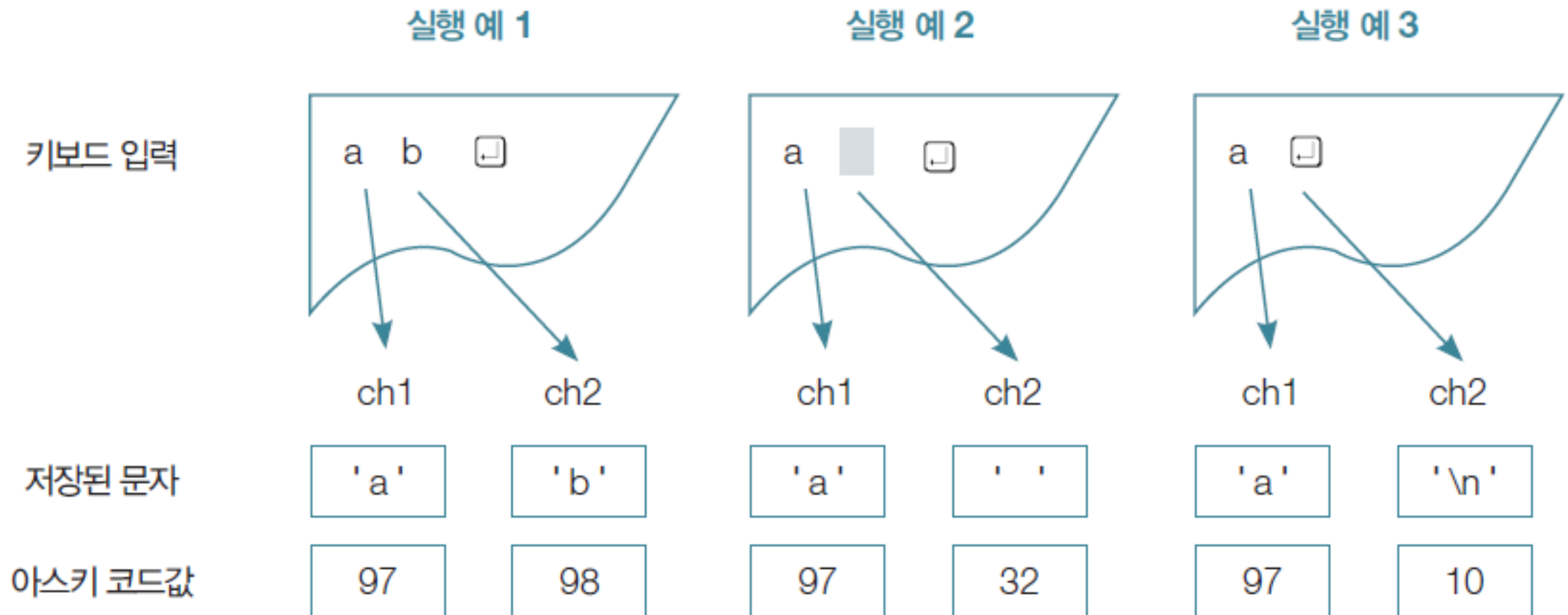
a

[a

]

SCANF 함수를 사용한 문자 입력

▶ 세 가지 예시 실행 후 메모리 상황



SCANF 함수를 사용한 문자 입력

- ▶ 제어 문자를 %c로 출력
 - ▶ 그 기능을 수행하므로 문자의 존재는 보이지 않음
- ▶ %d 써서 각 변수에 저장된 문자의 아스키 코드값 출력하면 입력된 문자를 명확히 확인 가능
 - ▶ 공백문자는 아스키 코드값 32, 개행문자는 10 출력

```
printf("%d, %d", ch1, ch2); // 세 번째 실행 예의 출력 결과는 97, 10
```

- ▶ 여러 개의 문자 입력할 때 데이터 구분 위해 칸 띄우거나 탭 키 사용
 - ▶ 공백문자나 탭문자가 데이터로 입력될 수 있음
- ▶ cf) scanf 함수가 제공하는 특별한 기능
 - ▶ %c 앞에 공백 사용하면 문자도 분리하여 가능

SCANF 함수를 사용한 문자 입력

- ▶ 스페이스, 탭, 엔터 키 눌렀을 때 입력되는 문자 - 화이트 스페이스 (white space)
 - ▶ %d, %lf, %s와 같은 변환문자열로 숫자나 문자열을 입력할 때는 데이터 구분하는 용도
 - ▶ 그 자체가 데이터로 입력되지는 않음
 - ▶ ex) 두 개의 int형 변수에 정수 동시에 입력하는 경우

```
int a, b;           // 두 개의 int형 변수
scanf("%d%d", &a, &b); // 두 변수에 동시 입력
```

SCANF 함수를 사용한 문자 입력

- ▶ scanf 함수가 호출되어 키보드로 '10(공백)20'과 같이 입력
 - ▶ 중간에 있는 공백문자는 10과 20을 구분하는 용도로만 쓰임 (공백은 숫자가 될 수 없기 때문)
- ▶ %c는 문자 입력 - 화이트 스페이스도 입력 대상
 - ▶ 화이트 스페이스 제외한 문자들만 입력하고 싶다면 %c 앞에 화이트 스페이스 중 아무거나 하나 추가
 - ▶ 공백 사용하는 것이 가장 쉬우므로 보통 한 칸 띄움

```
scanf(" %c %c", &ch1, &ch2);
```

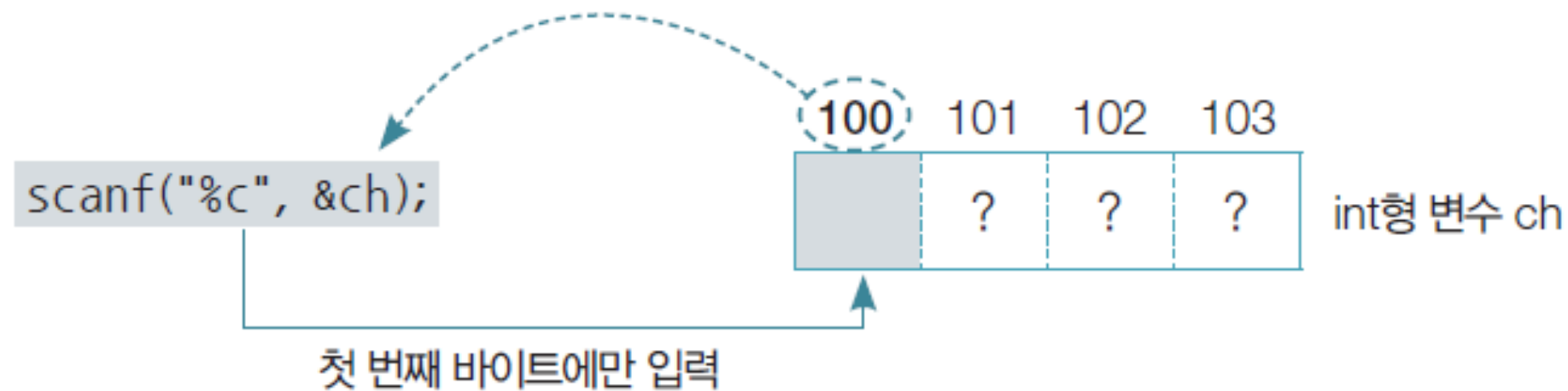
%c 앞에 화이트 스페이스 사용
(탭문자 \t나 개행문자 \n도 사용 가능)

SCANF 함수를 사용한 문자 입력

- ▶ ex) 'a(공백)b'와 같이 입력
 - ▶ ch1에는 a가 입력되고, ch2에는 공백이 아닌 b 입력
 - ▶ 중간에 공백을 여러 개 넣어 입력해도 모두 무시
- ▶ scanf 함수로 문자 입력할 때 주의점
 - ▶ int형 변수 사용하면 처음 한 바이트에만 입력
 - ▶ 입력한 문자를 메모리의 한 바이트 공간에 저장하도록 설계
 - ▶ int형 변수 사용하면 나머지 세 바이트에 있는 쓰레기값으로 인해 입력한 문자의 아스키 코드값 바로 사용 불가

SCANF 함수를 사용한 문자 입력

.....



```
int ch;                // int형 변수 사용
scanf("%c", &ch);       // int형 변수에 문자 입력
printf("%d", ch);       // 입력된 문자의 아스키 코드값 출력
```

실행
결과 A

-858993599 // 쓰레기값으로 인해 문자 A의 아스키 코드값이 출력되지 않음

SCANF 함수를 사용한 문자 입력

- ▶ scanf 함수로 문자 입력할 때 주의점
 - ▶ 입력한 int형 변수를 별도의 char형 변수에 대입
 - ▶ 입력한 문자만 사용하도록
 - ▶ 문자를 입력하는 경우는 가능한 char형 변수 사용


GETCHAR 함수와 PUTCHAR 함수

- ▶ 문자만 입출력 하는 경우
 - ▶ 문자 전용 함수를 쓰는 것이 효율적
 - ▶ `int getchar() ;` 매개변수가 없고 입력한 문자 반환
 - ▶ `int putchar(int) ;` 출력할 문자 인수로 줌

GETCHAR 함수와 PUTCHAR 함수

예제 11-4 getchar와 putchar 함수 사용

```
1. #include <stdio.h>
2.
3. int main(void)
4. {
5.     int ch;                // 입력 문자를 저장할 변수
6.
7.     ch = getchar();        // 함수가 반환하는 문자를 바로 저장
8.     printf("입력한 문자 : ");
9.     putchar(ch);          // 입력한 문자 출력
10.    putchar('\n');         // 개행문자 출력
11.
12.    return 0;
13. }
```

실행 결과 A 
입력한 문자 : A

GETCHAR 함수와 PUTCHAR 함수

▶ getchar 함수

- ▶ 매개변수가 없으므로 괄호만 사용하여 호출
- ▶ 호출된 함수는 키보드 입력한 문자 아스키 코드값 반환
 - ▶ 반환값 받을 변수도 int형 변수 사용
- ▶ 아스키 코드값은 1바이트로 모두 표현 가능
 - ▶ char형 변수 - 저장된 데이터가 문자임을 표현
 - ▶ 보기도 좋고 저장 공간도 절약

GETCHAR 함수와 PUTCHAR 함수

▶ putchar 함수

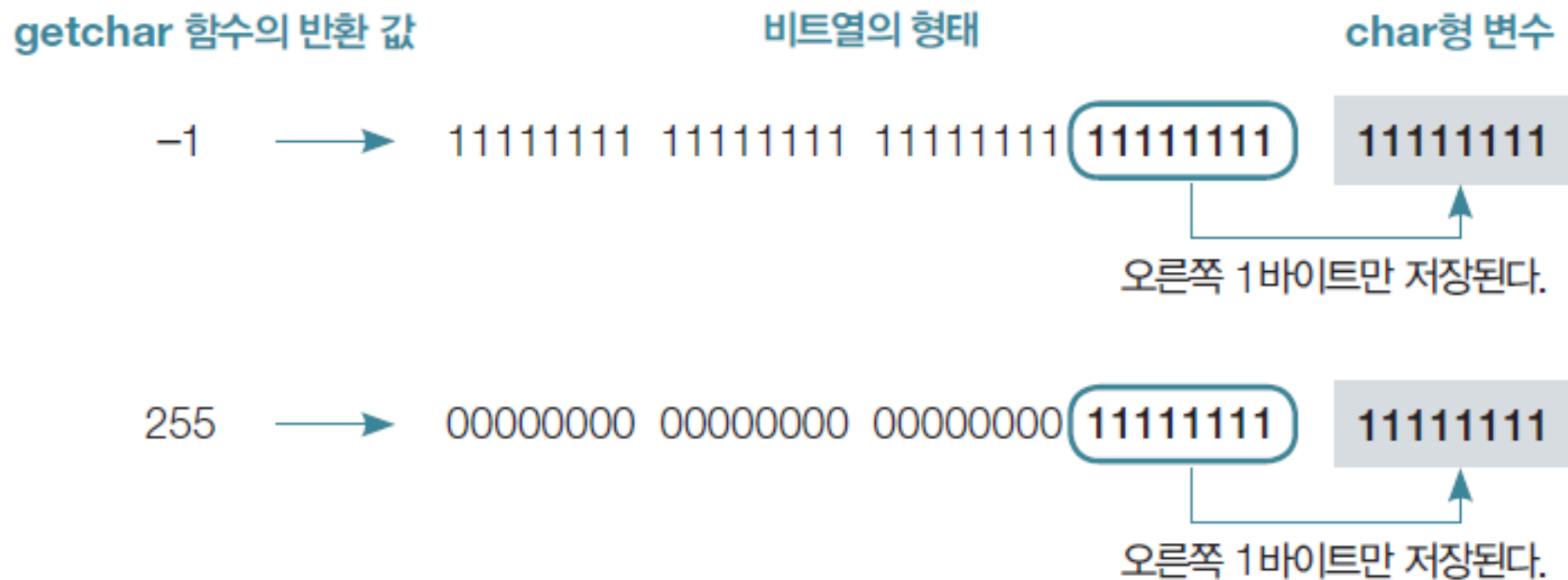
- ▶ 문자 상수나 문자의 아스키 코드값 인수로 주면 화면에 해당 문자 출력
- ▶ 출력한 문자 다시 반환하며 출력 과정에서 오류가 발생시 -1 반환

GETCHAR 함수와 PUTCHAR 함수

- ▶ getchar 함수의 반환형이 int형인 이유?
 - ▶ 문자 이외의 값도 반환하기 때문
 - ▶ 문자 입력 끝내기 위해 Ctrl+Z 키를 누르면 -1 반환
 - ▶ 이 값을 문자와 정확히 구분하기 위해 반환형으로 int형 사용
- ▶ 키보드에서만 입력한다면 반환형이 char형이라도 반환되는 문자와 -1 가능
 - ▶ 키보드에서는 아스키문자만 입력 가능
 - ▶ 아스키 코드값 범위는 0~127
- ▶ 데이터를 입력하는 경로가 파일
 - ▶ 반환하는 값이 255라면 파일의 데이터를 모두 읽은 경우 반환하는 -1과 구분이 불가능할 수 있음

GETCHAR 함수와 PUTCHAR 함수

- ▶ getchar 함수의 반환형이 int형인 이유?
 - ▶ -1은 모든 비트가 1인 상태로 저장
 - ▶ 1바이트 크기의 공간에서는 255와 -1을 표현하는 비트열 동일



버퍼를 사용하는 입력 함수

❖ 버퍼

- ▶ 메모리에 있는 임시 저장 공간
- ▶ 입출력 버퍼의 이름 직접 사용하는 함수

표 11-3 입출력 버퍼의 이름을 직접 사용하는 함수

구분	함수 사용법	기능
입력	<code>int ch;</code> <code>ch = fgetc(stdin);</code>	int형 변수에 입력 공백문자, 탭문자, 개행문자도 입력 입력 문자의 아스키 코드값 반환 입력 버퍼 stdin 사용
출력	<code>fputc(ch, stdout);</code>	문자 출력 전용함수, 출력할 문자 전달 출력할 문자와 출력 버퍼 stdout 사용
버퍼 관리	<code>fflush(stdin);</code> <code>fflush(stdout);</code>	입력 버퍼의 내용 삭제 출력 버퍼의 내용 화면에 출력

SCANF 함수가 문자를 입력하는 과정

➤scanf 함수

- 키보드 이전에 버퍼로부터 데이터 입력

- 버퍼는 프로그램의 실행 중 운영체제가 자동 할당하는 저장 공간

 - 최초 입력할 때 필요한 데이터를 한꺼번에 저장

- 키보드로 입력하는 데이터는 일단 버퍼에 저장된 후 scanf 함수에 의해 변수에 입력

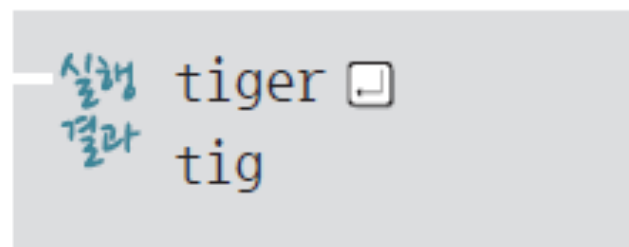
 - 다음 scanf 함수는 호출 즉시 버퍼에서 데이터 가져옴

SCANF 함수가 문자를 입력하는 과정

.....

예제 11-5 버퍼를 사용하는 문자입력

```
1. #include <stdio.h>
2.
3. int main(void)
4. {
5.     char ch;
6.     int i;
7.
8.     for(i = 0; i < 3; i++) // 세 번 반복
9.     {
10.         scanf("%c", &ch); // 문자 입력
11.         printf("%c", ch); // 입력된 문자 출력
12.     }
13.
14.     return 0;
15. }
```



```
실행 결과 tiger
tig
```

SCANF 함수가 문자를 입력하는 과정

- ▶ 두 번째 scanf 함수 호출부터는 버퍼에 남아 있는 문자열에서 차례로 다음 문자 가져옴
- ▶ 새로운 키보드 입력 필요 없음
- ▶ scanf 함수가 버퍼에 저장된 데이터를 모두 가져온다면 키보드에서 추가로 데이터 입력



SCANF 함수가 문자를 입력하는 과정

- ▶ scanf 함수가 입력하지 않은 데이터 버퍼에 남긴 채 프로그램 종료
 - ▶ 다음 입력에 사용하거나 불필요하면 지울 수도
 - ▶ 버퍼의 내용을 지울 때는 fflush 함수
- ▶ 데이터의 입력 방식
 - ▶ 입력 데이터는 엔터 치는 순간 버퍼에 저장
 - ▶ 개행문자도 함께 저장
 - ▶ 버퍼에 있는 개행문자도 하나의 데이터로 입력 가능

SCANF 함수가 문자를 입력하는 과정

- ▶ ex) 개행문자가 나올 때까지 문자를 반복적으로 입력 후
 - ▶ 출력하면 키보드로 입력한 한 줄 데이터를 길이와 상관없이 화면에 출력

```
while(1)
{
    scanf("%c", &ch);           // 버퍼에서 한 문자 입력받음
    if(ch == '\n') break;       // 개행문자인 경우 반복 종료
    printf("%c", ch);           // 입력한 문자 출력
}
```

SCANF 함수 반환값 활용

- ▶ 프로그램 사용자가 키보드로 한 줄 입력 시 입력 끝내려면 엔터 키
 - ▶ 개행문자 또한 하나의 입력 데이터로 쓰면 입력 종료하는 별도 신호 필요
 - ▶ scanf 함수 반환값 사용
 - ▶ 키보드로 Ctrl+Z를 누르면 -1 반환
 - ▶ 유닉스나 리눅스 시스템에서는 Ctrl+D 사용
 - ▶ scanf 함수가 -1 반환하기 전까지 반복 입력하면 개행문자 포함한 모든 문자 데이터로 사용

SCANF 함수 반환값 활용

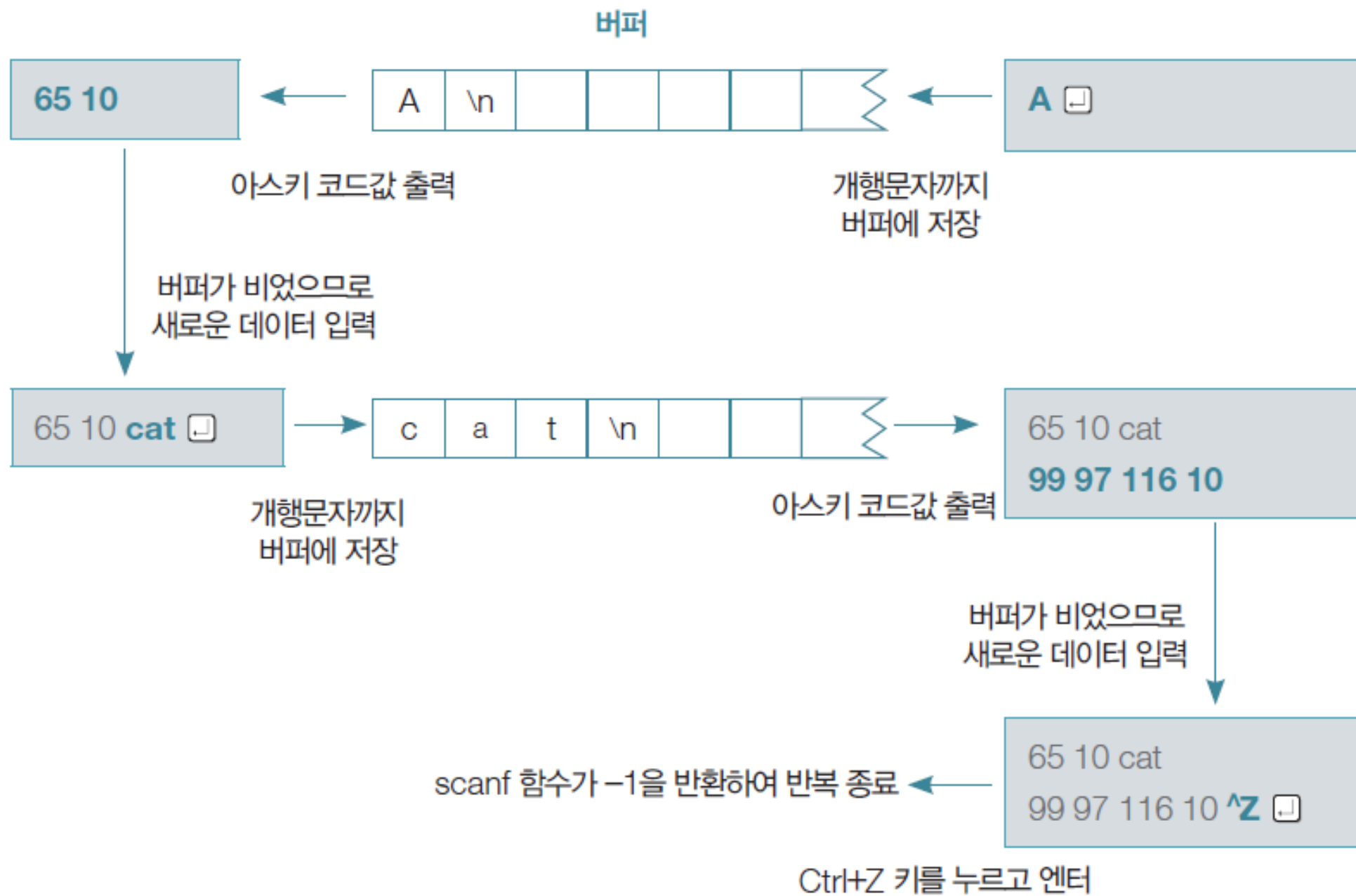
예제 11-6 입력 문자의 아스키 코드값을 출력하는 프로그램

```
1. #include <stdio.h>
2.
3. int main(void)
4. {
5.     int res;                // scanf 함수의 반환값을 저장할 변수
6.     char ch;                // 문자를 입력할 변수
7.
8.     while(1)
9.     {
10.         res = scanf("%c", &ch); // 문자 입력, Ctrl+Z를 누르면 -1 반환
11.         if(res == -1) break;    // 반환값이 -1이면 반복 종료
12.         printf("%d ", ch);    // 입력된 문자의 아스키 코드값 출력
13.     }
14.
15.     return 0;
16. }
```

실행 결과

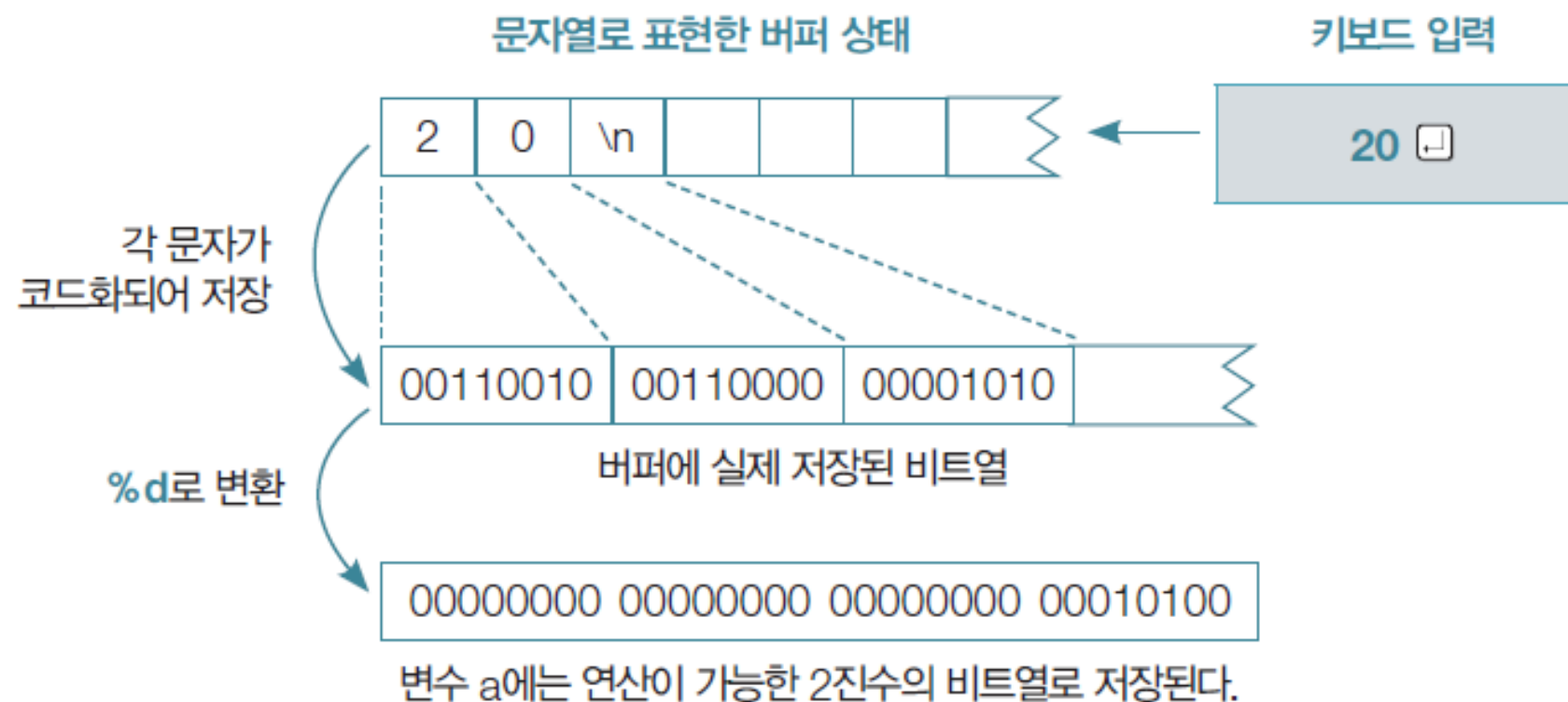
```
A
65 10 cat
99 97 116 10 ^Z
```

SCANF 함수 반환값 활용



SCANF 함수 반환값 활용

- ▶ 키보드로 숫자를 입력하는 경우
 - ▶ 일단 문자열의 형태로 버퍼에 저장
 - ▶ 그 후에 문자열이 실제 연산이 가능한 값으로 변환되어 변수에 저장
 - ▶ ex) int형 변수 a에 20 입력하는 경우 문자 '2'와 '0'이 각각 아스키 코드값으로 코드화되어 버퍼에 저장
 - ▶ 그 후 변환문자의 지시에 따라 연산이 가능한 숫자로 변환되어 변수에 저장



SCANF 함수 반환값 활용

- ▶ 키보드로 숫자를 입력하는 경우
 - ▶ 변환문자는 코드화된 문자열을 숫자로 변환하는 방법을 scanf 함수에 알려주는 역할
- ▶ 같은 입력에 대해 %lf 변환문자를 쓰고 실수형 변수에 입력
 - ▶ 버퍼에 저장된 상태는 같지만 IEEE 754 표준에 따라 변환
 - ▶ 변수에 저장되는 비트열의 크기와 형태는 달라짐

SCANF 함수 반환값 활용

- ▶ 키보드로 숫자를 입력하는 경우
 - ▶ scanf 함수의 반환값과 비교하는 값으로 -1 대신 EOF
 - ▶ stdio.h 헤더 파일에 소스코드에 있는 EOF라는 이름을 -1로 바꾸는 전처리 지시자 존재
 - ▶ EOF는 End Of File의 뜻이므로 -1 대신에 입력의 끝을 의미하는 이름으로 사용하면 좀 더 읽기 쉬운 코드 작성

```
res = scanf("%c", &ch);           // scanf 함수의 반환값을 res에 저장
if(res == EOF) break;             // EOF는 -1로 바뀌므로 결국 res와 -1을 비교한다.
```

GETCHAR 함수를 사용한 문자열 입력

.....

예제 11-7 getchar 함수를 사용한 문자열 입력

```
1. #include <stdio.h>
2.
3. void my_gets(char *str, int size);
4.
5. int main(void)
6. {
7.     char str[7];                // 문자열을 저장할 배열
8.
9.     my_gets(str, sizeof(str));   // 한 줄의 문자열을 입력하는 함수
10.    printf("입력한 문자열 : %s\n", str); // 입력한 문자열 출력
11.
12.    return 0;
13. }
14.
```

GETCHAR 함수를 사용한 문자열 입력

```
15. void my_gets(char *str, int size)           // str은 char 배열, size는 배열의 크기
16. {
17.     char ch;                                // getchar 함수의 반환값을 저장할 변수
18.     int i = 0;                              // str 배열의 첨자
19.
20.     ch = getchar();                          // 첫 번째 문자 입력
21.     while((ch != '\n') && (i < size-1))      // 배열의 크기만큼 입력
22.     {
23.         str[i] = ch;                        // 입력한 문자를 배열에 저장
24.         i++;                                // 첨자 증가
25.         ch = getchar();                    // 새로운 문자 입력
26.     }
27.     str[i] = '\0';                          // 입력된 문자열의 끝에 널문자를 저장
28. }
```

실행
결과

[실.행.결.과 1]

a boy

입력한 문자열 : a boy

[실.행.결.과 2]

Be happy!

입력한 문자열 : Be hap

GETCHAR 함수를 사용한 문자열 입력

- ▶ getchar 함수 사용

- ▶ 키보드로 입력한 한 줄 문자열을 char 배열로 저장

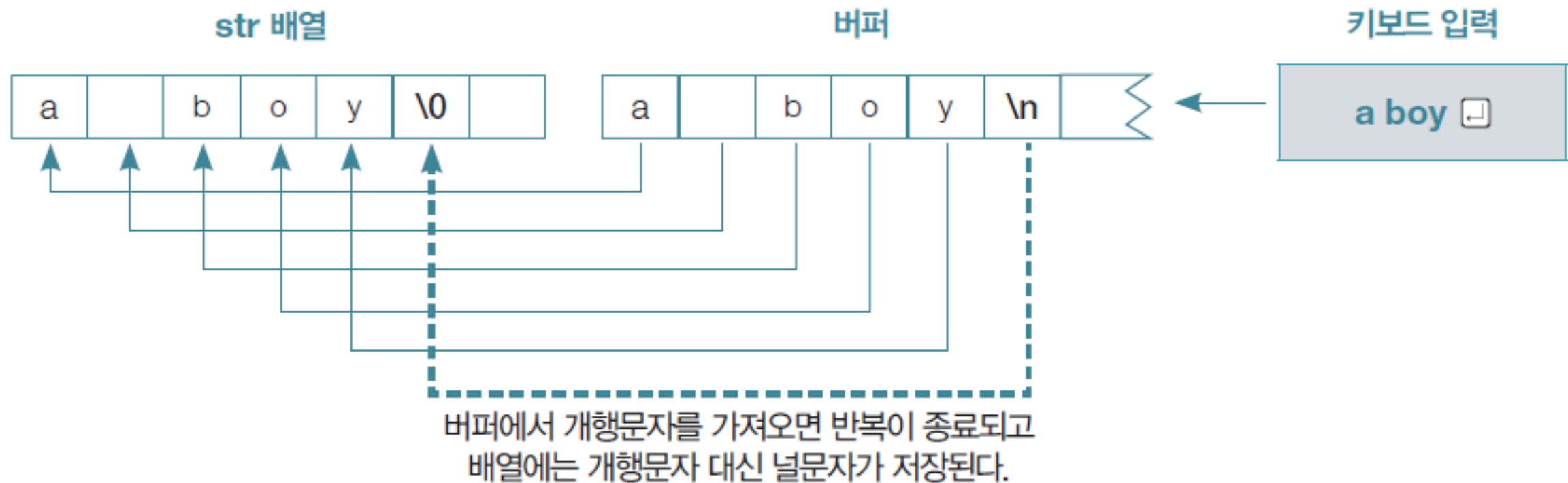
- ▶ 배열의 크기 넘는 문자열 입력한 경우

- ▶ 배열의 크기만큼만 입력하도록 작성 (할당되지 않은 메모리 침범하지 않도록 작성)

- ▶ 크기가 다른 배열에도 사용할 수 있도록 함수로 작성

GETCHAR 함수를 사용한 문자열 입력

- ▶ 키보드로 한 줄의 데이터를 모두 입력하여 버퍼에 저장
- ▶ getchar 함수 반복 사용하여 버퍼로부터 문자 하나씩 가져와 배열에 차례로 저장



FFLUSH 함수

▶ scanf와 getchar 함수는 같은 버퍼 사용

▶ 입력 데이터 공유

▶ 앞서 실행한 입력 함수가 버퍼에 남겨둔 데이터를 그 이후에 수행되는 함수가 잘못 가져갈 가능성 있음

▶ 버퍼에 남아 있는 불필요한 데이터는 미리 제거

▶ 입력 버퍼의 내용을 지울 때는 fflush(stdin) 함수 사용

```
int fflush(FILE *stream)
```

FFLUSH 함수

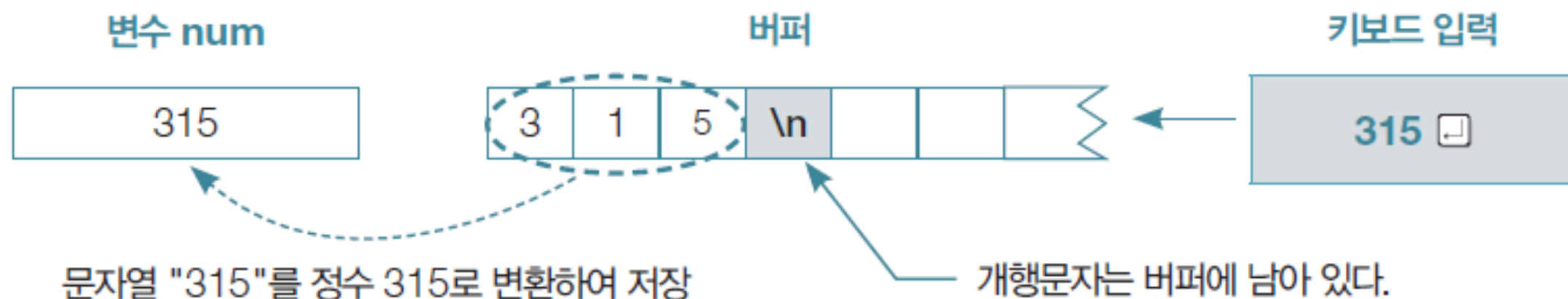
예제 11-8 fflush 함수가 필요한 경우

```
1. #include <stdio.h>
2.
3. int main(void)
4. {
5.     int num, grade;           // 학번과 학점을 저장할 변수
6.
7.     printf("학번 입력 : ");
8.     scanf("%d", &num);        // 학번 입력
9.     fflush(stdin);            // 버퍼에 남아 있는 개행문자 제거
9.     printf("학점 입력 : ");
10.    grade = getchar();         // 학점 입력
11.    printf("학번 : %d, 학점 : %c", num, grade);
12.
13.    return 0;
14. }
```

실행 결과
학번 입력 : 315
학점 입력 : A
학번 : 315, 학점 : A

FFLUSH 함수

- ▶ scanf 함수는 일단 버퍼로부터 입력 시도
 - ▶ 처음에는 버퍼가 비어 있으므로 키보드로부터 입력 받기 위해 대기
 - ▶ 315 입력하고 엔터 키를 치면 315와 개행문자가 함께 버퍼에 저장
 - ▶ 문자열 315는 정수로 변환되어 변수 num에 저장
 - ▶ 버퍼에는 개행문자만 남음
- ▶ 남은 개행문자는 10행에서 하나의 문자 입력하는 getchar 함수에 의해 읽혀짐
 - ▶ 이후 fflush 함수로 개행문자 제거



FFLUSH 함수

- ▶ 그 이후에는 버퍼가 비어 있음
 - ▶ 다시 키보드로부터 입력 시도하여 학점을 제대로 입력
- ▶ 만약 fflush 함수를 사용하지 않으면?
 - ▶ scanf 함수가 학번을 입력할 때 버퍼에 남아 있던 개행문자를 그 이후 호출되는 getchar 함수가 가져가게 됨
 - ▶ 학점을 추가로 입력 받지 못하고 프로그램이 종료
 - ▶ grade에 저장된 개행문자가 출력되어 줄 바뀜

FFLUSH 함수

▶ fflush 함수를 사용하지 않을 때의 출력 결과

학번 입력 : 315 ☐

학점 입력 : 학번 : 315, 학점 : // 학점을 입력할 수 없고 학점을 출력할 곳에서 줄이 바뀜

FFLUSH 함수

- ▶ fflush 함수의 인수로 주는 stdin
 - ▶ standard input의 의미로 표준 입력장치인 키보드와 연결된 버퍼 이름
- ▶ stdout은 출력 버퍼 이름
 - ▶ printf나 putchar와 같은 표준 출력 함수들이 공유
- ▶ fflush 함수를 출력 버퍼에 사용하면 버퍼의 내용을 즉시 모니터로 출력

FFLUSH 함수

- ▶ 문자 입출력 함수에는 fgetc나 fputc 함수와 같이 stdin과 stdout을 직접 표시하는 경우도 있음

```
int ch;  
ch = fgetc(stdin);      // 키보드로 입력한 문자를 반환합니다.  
fputc(ch, stdout);      // 주어진 문자를 화면에 출력합니다.
```
