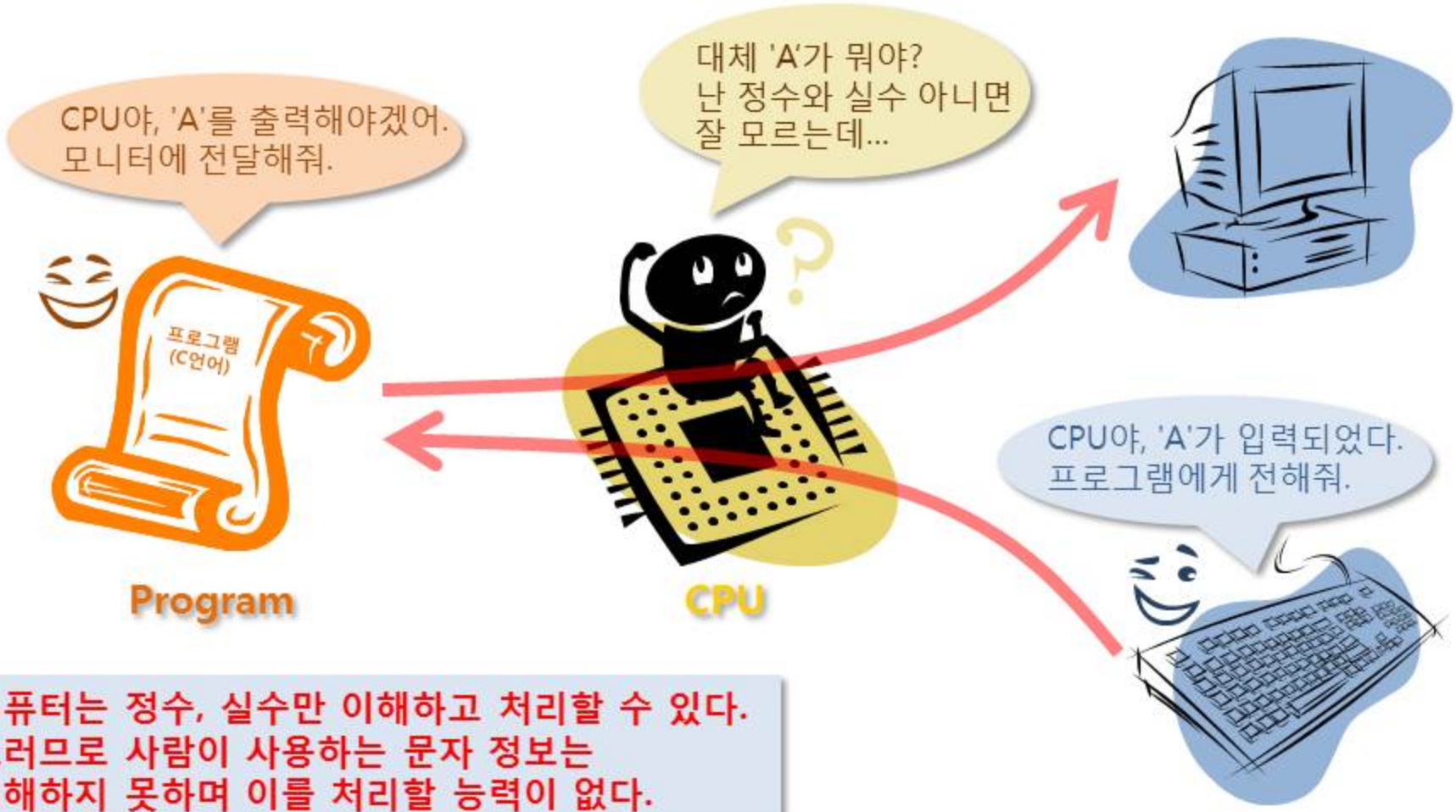
C언어 강의자료

문정욱

C언어 더 알아보기 1

문자 코드

문자 정보의 입력과 출력



컴퓨터는 정수, 실수만 이해하고 처리할 수 있다.
그러므로 사람이 사용하는 문자 정보는
이해하지 못하며 이를 처리할 능력이 없다.

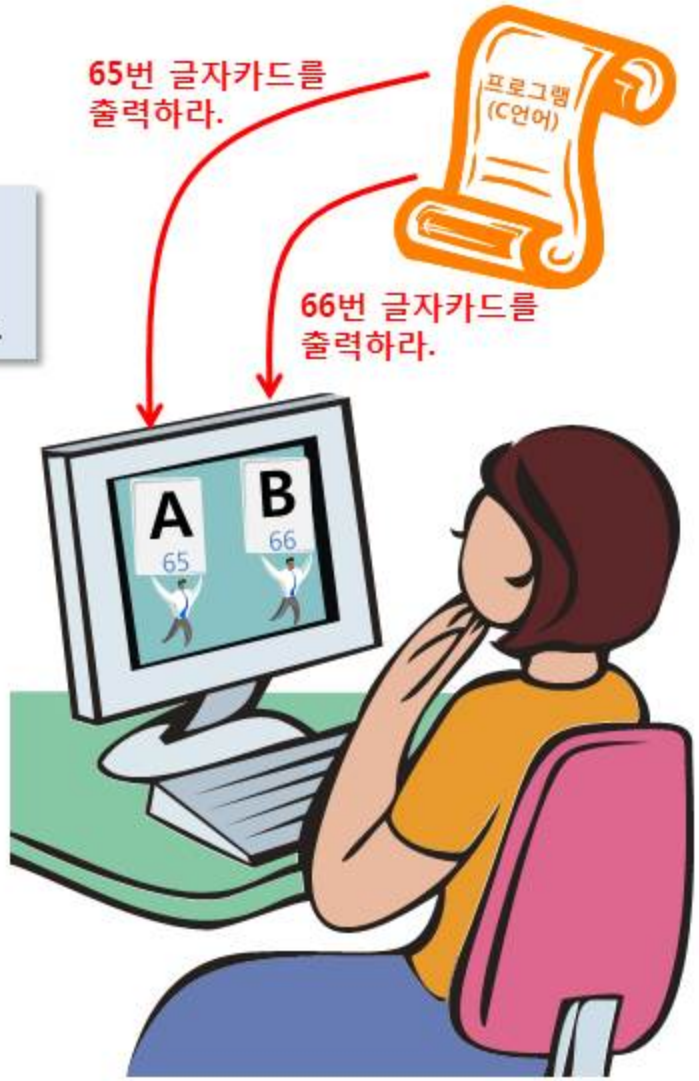
문자 코드

문자 정보의 출력 과정

컴퓨터는 모니터에 문자 정보를 출력해야 할 때
사용자에게 문자 정보 대신
고유 번호(code)가 적혀있는 **글자 카드(font)**를 보여준다.

컴퓨터는 고유 번호를 사용하여
글자 카드를 구별할 수 있다.
하지만, 글자카드에 적혀있는
문자정보의 의미를 전혀 이해하지 못한다.

**사람들이 보기에는 컴퓨터가 문자 정보 자체를
이해하고 처리할 수 있는 것처럼 보이지만
실제는 글자 카드의 번호(정수)를 전달하고 처리하는
것이다.**



문자 코드

문자 정보의 입력 과정

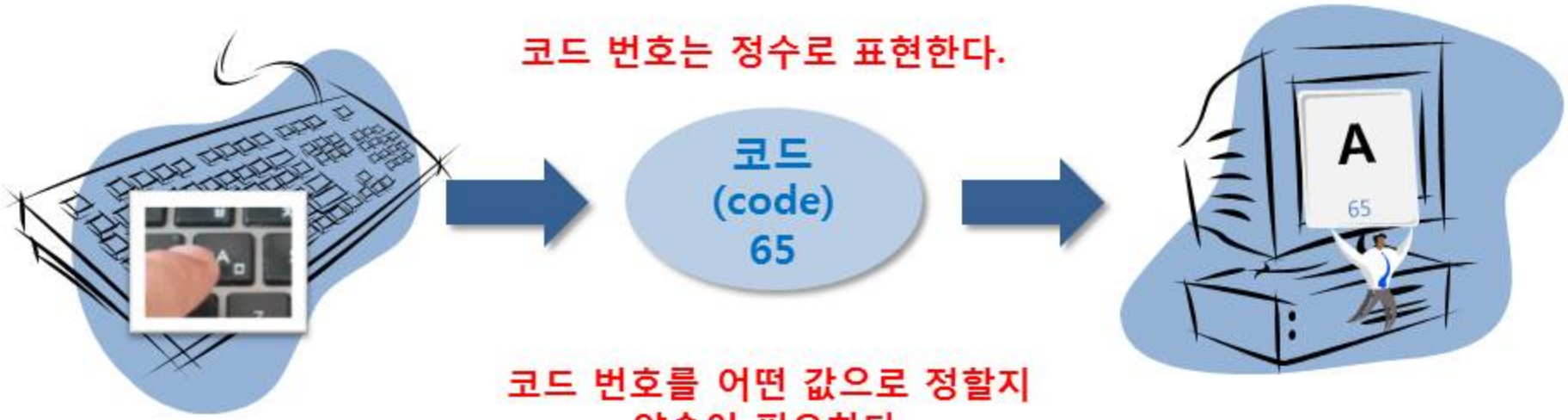
문자 정보를 입력 받을 때에도 같은 방식을 사용한다.

컴퓨터는 문자 정보를 입력 받기 위해
키보드로부터 문자 정보 자체를 입력 받는 것이 아니라
입력된 글쇠의 번호(정수)를 입력 받는다.

문자 코드

문자 정보를 교환하기 위한 코드(code)

문자 정보를 모니터로 출력하기 위해 글자 카드(font)의 고유 번호(code)를 전달한다.
문자 정보를 키보드로부터 입력 받을 때 글쇠(key)의 고유 번호(code)를 전달받는다.



문자 코드

ASCII(American Standard Code for Information Interchange)

16진법

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

'A' == $41_{(16)}$ == $65_{(10)}$

'z' == $7A_{(16)}$ == $122_{(10)}$

문자 코드

ASCII Code (7 bits)

총 128개의 문자

$2^7 = 128$ → 7비트 필요

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3 숫자	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4 알파벳 대문자	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5 알파벳 대문자	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6 알파벳 소문자	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7 알파벳 소문자	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

제어문자

key	name	symbol	usage
NUL	Null	'\0'	String termination char
BS	Back space	'\b'	Erase a previous char
HT	Tab	'\t'	Column alignment
LF	New line	'\n'	Line feed

문자 코드

- 문자 코드의 출력
  - printf 에서 %c 의미
    - **문자 코드(정수)**에 해당하는 **글자 카드(font)**를 화면을 통해 보여줘라.

	0	1	2	3	
0	NUL	SOH	STX	ETX	E
1	DLE	DC1	DC2	DC3	[
2		!	"	#	
3	0	1	2	3	
4	@	A	B	C	
5	P	Q	R	S	
6	`	a	b	c	
7	p	q	r	s	

'A' == $41_{(16)}$ == $65_{(10)}$

```
#include <stdio.h>

int main(void)
{
    printf("%c\n", 65);
    return 0;
}
```

65번 글자 카드(font)를 출력하라.

65번 글자 카드(font) 출력

입출력 결과

```
A
계속하려면 아무 키나 누르십시오 . . .
```

문자 코드

- 문자 코드의 저장
  - ASCII 코드는 **7bit 정수**이다.
    이는 **정수형 변수**에 저장할 수 있다.
  - ASCII 코드를
    int 타입(32비트=4바이트)의
    변수에 저장할 경우
    ☞ 7bit 정보 이외에 나머지
    25비트의 저장공간이 낭비된다.
  - **메모리 낭비를 줄이기** 위해
    ASCII 코드를 저장할
    정수 타입이 필요
    ☞ **1바이트 정수 타입이 적당**

```
#include <stdio.h>

int main(void)
{
    int  i = 65;

    printf("%c\n", i);
    return 0;
}
```

대문자 A를 의미하는 문자 코드 65가 변수 i에 저장됨.

입출력 결과

```
A
계속하려면 아무 키나 누르십시오 . . .
```

문자형 변수

- 문자형 변수의 사용
  - **char : 1byte(8bits) 정수형**
    - *주의: 문자형 변수에는 문자가 저장되는 것이 아니라 문자 코드(정수)가 저장된다.*

```
#include <stdio.h>

int main(void)
{
    char ch = 65;
    printf("%c\n", ch );
    return 0;
}
```

대문자 A를 의미하는 문자 코드 65가 변수 ch에 저장됨.

character의 약자

ch 0100 0001

입출력 결과

```
A
계속하려면 아무 키나 누르십시오 . . .
```

문자형 변수

- 문자 코드의 표현 방식
  - 10진 정수: 0~9 사용
  - 16진 정수: 0X로 시작
  - 문자 정수: 작은 따옴표(') 사용

	0	1	2	3
0	NUL	SOH	STX	ETX
1	DLE	DC1	DC2	DC3
2		!	"	#
3	0	1	2	3
4	@	A	B	C
5	P	Q	R	S
6	`	a	b	c
7	p	q	r	s

'A' == $41_{(16)}$ == $65_{(10)}$

같은 의미

```
#include <stdio.h>

int main(void)
{
    char ch;

    ch = 65;
    printf("%c\n", ch );

    ch = 0x41;
    printf("%c\n", ch );

    ch = 'A';
    printf("%c\n", ch );
    return 0;
}
```

16 진수 정수의 표현. 0X로 시작함. 16진수를 10진수로 변환 필요 없음.

대문자 A가 아니라 65(정수)를 의미. 작은 따옴표 사용. ASCII table을 찾아 볼 필요 없음.

입출력 결과

```
A
A
A
계속하려면 아무 키나 누르십시오 . . .
```

문자 코드

- 문자 코드의 연산
  - 작은 따옴표에 있는 문자를 문자 정보로 인식하면 안되고 **정수**로 인식해야 한다.

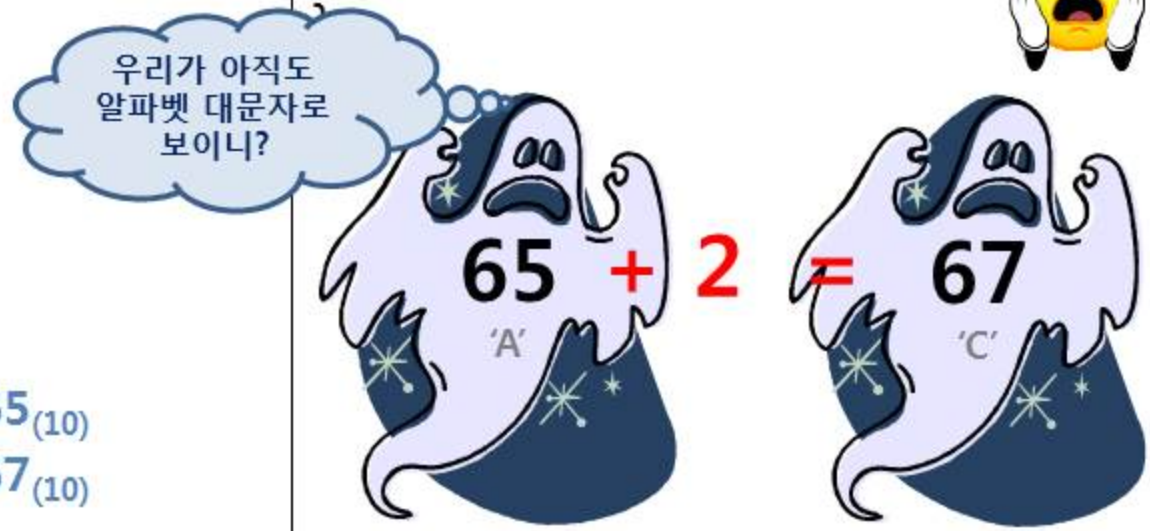
	0	1	2	3	
0	NUL	SOH	STX	ETX	E
1	DLE	DC1	DC2	DC3	[
2		!	"	#	
3	0	1	2	3	
4	@	A	B	C	
5	P	Q	R	S	
6	`	a	b	c	
7	p	q	r	s	

'A' == $41_{(16)}$ == $65_{(10)}$

'C' == $43_{(16)}$ == $67_{(10)}$

```
#include <stdio.h>

int main(void)
{
    char ch = 'A' + 2;  // ch==67

    printf("%d\n", ch);
    printf("%c\n", ch);
    return 0;
}
```

문자 A에 2를 더하라고?
65에 2를 더하라는 뜻!!!



문자형 변수

- 문자 코드의 저장
  - **문자 코드는** 정수이므로 **모든 정수형 변수에 저장 가능**하다.
  - 다만 정수형 변수의 메모리 크기가 너무 크면 메모리 공간이 낭비될 수 있다.

```
#include <stdio.h>

int main(void)
{
    int  i = 'A';  // i == 65
    char c = 'A';  // c == 65


    printf("%d\n", i); // ok
    printf("%d\n", c); // ok
                       변수에 저장된 값을
                       10진수로 출력

    printf("%c\n", i); // ok
    printf("%c\n", c); // ok
    return 0;          변수에 저장된 값에 해당하는
                       글자 카드(font)를 출력
}
```

입출력 결과

```
65
65
A
A
계속하려면 아무 키나 누르십시오 . . .
```

10진수로 출력

글자 카드를 출력

문자형 변수

- 문자 코드의 입력
  - scanf 에서 %c 의미
    - 문자를 입력하면 그 문자에 해당하는 문자 코드 값(정수)를 변수에 저장한다.

```
#include <stdio.h>

int main(void)
{
    char ch; // use char type

    scanf("%c", &ch );

    printf("%c\n", ch );
    printf("%d\n", ch );
    return 0;
}
```

입출력 결과

```
A
A
65
계속하려면 아무 키나 누르십시오 . . .
```

문자형 변수

- 정수 및 문자 코드 입력 시 scanf 사용시 주의 점
  - %d : int 형 변수 사용
    - char 형 변수 사용하면 실행오류 (run-time error) 발생
  - %c : char 형 변수 사용
    - int 형 변수 사용하면 논리오류 (logical error) 발생 가능
      ☞ 변수 초기화의 여부, CPU의 종류에 따라 오류발생 여부가 달라진다.

```c
#include <stdio.h>

int main(void)
{
    char c = 97;        // 97 means 'a'
    int  i = 97;

    printf("%c\n", c); // ok
    printf("%c\n", i); // ok

    printf("%d\n", c); // ok
    printf("%d\n", i); // ok

    scanf("%c", &c);   // ok
    scanf("%c", &i);   // logical error

    scanf("%d", &c);   // run-time error
    scanf("%d", &i);   // ok
    return 0;
}
```

변수

정수형 (integer type)

16비트 컴퓨터에서 int의 크기는 16비트
32비트 컴퓨터에서 int의 크기는 32비트
이 표는 32비트 컴퓨터 기준

Type	Long Name	Size	Min Value	Max Value
char	char	1 byte (8bits)	-2^7	2^7-1
short	short int	2 bytes (16bits)	-2^{15}	$2^{15}-1$
int	int	4 bytes (32bits)	-2^{31}	$2^{31}-1$
long	long int	4 bytes (32bits)	-2^{31}	$2^{31}-1$
long long	long long int	8 bytes (64bits)	-2^{63}	$2^{63}-1$

- 변수의 구성요소: 주소, 값
- 값의 구성요소: 이진자료, 타입
- 타입의 구성요소: 값의 표현방식,
  메모리 공간의 크기

값의 표현방식은
- 양수: 2진수
- 음수: 2의 보수

값의 표현방식은 동일하고
메모리 공간의 크기만 다르다.

문자 코드

- 백 스페이스(BS, back space)
  - 키보드의 Back space key와 같은 역할
    - 앞 출력된 문자 한 개를 삭제한다.

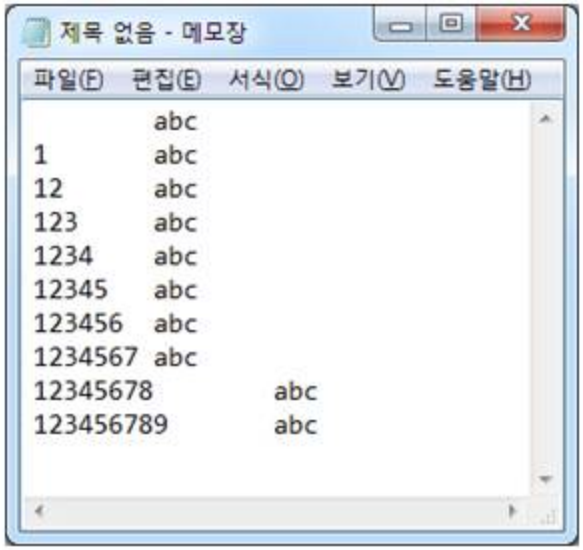
```
#include <stdio.h>

int main(void)
{
    printf("abcd\befg\n");
    printf("abcd\b\befg\n");
    return 0;
}
```

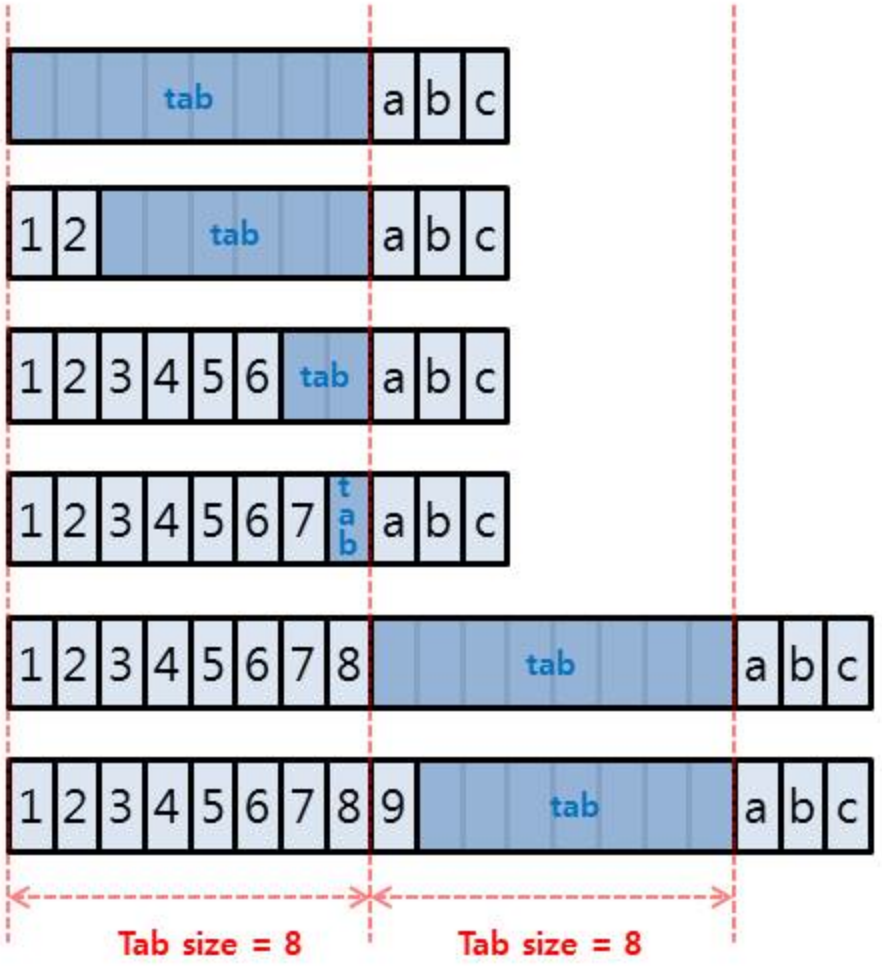
입출력 결과

```
abcefg
abefg
계속하려면 아무 키나 누르십시오 . . .
```

문자 코드

탭 문자의 크기 조절





문자 코드

- 탭(HT, Horizontal Tab)
  - 키보드의 tab key와 같은 역할
    - 앞 출력된 문자의 개수와 tab 문자 크기의 합이 8의 배수가 되도록 tab 문자의 크기를 1~8로 조절함.
  - 도표 작성 용도

※ tab key = tabulator key

```c
#include <stdio.h>

int main(void)
{
    printf("abcd\tefg\n");
    printf("ab\t124\tfg\n");
    return 0;
}
```

입출력 결과

```
abcd    efg
ab      124     fg
계속하려면 아무 키나 누르십시오 . . .
```

문자 코드 입출력 함수

- 입력 함수

```
int getchar(void);
```

  - 키보드로 입력 받은 문자 정보를 ASCII 코드로 반환한다.
  - 반환 값의 타입은 int이다.
  - 입력이 종료될 경우 EOF(-1, (end of file)을 반환한다.

- 출력 함수

```
void putchar(int ch);
```

  - 전달 받은 ASCII 코드에 해당하는 문자를 화면에 출력한다.

```
#include <stdio.h>

int main(void)
{
    int c;          // use int type.

    c = getchar(); // char input
    putchar(c);    // char output
    return 0;
}
```

입출력 결과

```
a
a계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
b
b계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
^Z
 계속하려면 아무 키나 누르십시오 . . .
```

-1은 ASCII 코드에 정의되어있지 않아서 제대로 출력 안됨.

문자 코드 출력 함수

putchar을 사용한 문자 코드 출력

```
#include <stdio.h>

int main(void)
{
    int ch = 65;

    putchar(ch);
    return 0;
}
```

입출력 결과

```
A계속하려면 아무 키나 누르십시오 . . .
```

printf를 사용한 문자 코드 출력

```
#include <stdio.h>

int main(void)
{
    int ch = 65;

    printf("%c", ch);
    return 0;
}
```

입출력 결과

```
A계속하려면 아무 키나 누르십시오 . . .
```

문자 코드 입력 함수

getchar을 사용한 문자 코드 입력

```
#include <stdio.h>

int main(void)
{
    int ch;

    ch = getchar();
    printf("%c", ch);
    return 0;
}
```

반드시 int 형 사용

입출력 결과

```
A
A계속하려면 아무 키나 누르십시오 . . .
```

scanf를 사용한 문자 코드 입력

```
#include <stdio.h>

int main(void)
{
    char ch;

    scanf("%c", &ch);
    printf("%c", ch);
    return 0;
}
```

반드시 char 형 사용

입출력 결과

```
A
A계속하려면 아무 키나 누르십시오 . . .
```

문자 코드의 입력

- EOF(end of file)의 입력
  - 키보드 입력 시 ctrl-z를 입력하면 더 이상 입력을 하지 않겠다는 의미

    ☞ ctrl-z 는 control key와 z를 동시에 눌러야 한다.
  - 이 경우 getchar 함수는 EOF(-1)을 반환한다.

입출력 결과

```
A
65
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
^Z
-1
계속하려면 아무 키나 누르십시오 . . .
```

```c
#include <stdio.h>

int main(void)
{
    int ch;

    ch = getchar();
    printf("%d\n", ch);
    return 0;
}
```

문자 입출력 함수

- ASCII code를 출력을 위한 printf() 함수의 FSF
  - %c : 해당 문자 카드(font) 출력
  - %d : 10진수로 출력
  - %x : 16진수로 출력

입출력 결과

```
a
input char is a
ascii code is 97
ascii code is 61(hexa)
계속하려면 아무 키나 누르십시오 . . .
```

```c
#include <stdio.h>

int main(void)
{
    int c;

    c = getchar();

    printf("input char is %c\n", c);
    printf("ascii code is %d\n", c);
    printf("ascii code is %x(hexa)\n", c);
    return 0;
}
```

scanf 동작 원리(문자코드의 입력)

- 값 앞에 WS가 있을 경우
  - **WS를 무시하지 않고**
    해당 WS의 문자코드를 저장

입출력 결과
```
abc
61 62 63 0A
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과
```
 bc
20 62 63 0A
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과
```
  c
20 20 63 0A
계속하려면 아무 키나 누르십시오 . . .
```

```c
#include <stdio.h>

int main(void)
{
    char ch;

    scanf("%c", &ch );
    printf("%02X ", ch );

    scanf("%c", &ch );
    printf("%02X ", ch );

    scanf("%c", &ch );
    printf("%02X ", ch );

    scanf("%c", &ch );
    printf("%02X\n", ch );
    return 0;
}
```

16진수(최소 2자리) 출력하되
상위 숫자가 0이면 그대로 0을 출력함.

scanf 동작 원리(문자코드의 입력)

- 값 앞에 WS가 있을 경우
  - **%c 앞에 공백(space)**을 두면 WS는 무시한 후 값을 저장

입출력 결과
```
abcd
61 62 63 64
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과
```
 abcd
61 62 63 64
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과
```
 a b cd
61 62 63 64
계속하려면 아무 키나 누르십시오 . . .
```

```c
#include <stdio.h>

int main(void)
{
    char ch;

    scanf(" %c", &ch );
    printf("%02X ", ch );

    scanf(" %c", &ch );
    printf("%02X ", ch );

    scanf(" %c", &ch );
    printf("%02X ", ch );

    scanf(" %c", &ch );
    printf("%02X\n", ch );
    return 0;
}
```

FSF 앞에 공백(space)가 들어감

문자열(string)

- 문자열의 의미
  - 문자열: 문자의 나열
  - 배열의 각 요소에 문자 코드를 저장함.
  - char 타입의 배열에 저장하기 좋음
  - **문자열 = char 타입의 배열** 이라고 봐도 무방함.
  - 문자열의 초기화는 배열 초기화와 같은 방식을 사용할 수 있음.

```
#include <stdio.h>

int main(void)
{
    char a[4] = { 97, 98, 99, 0 };
    char b[4] = { 'a', 'b', 'c', '\0' };

    return 0;
}
```

문자열(string)

- 문자열의 끝
  - 마지막에는 '\0' 저장

    ☞ **문자열의 마지막을 의미하는 표시** 일 뿐 전달하고자 하는 문자열 내용은 아니다.

	0	1	2	3
0	NUL	SOH	STX	ETX
1	DLE	DC1	DC2	DC3
2		!	"	#
3	0	1	2	3
4	@	A	B	C
5	P	Q	R	S
6	`	a	b	c
7	p	q	r	s

'\0' == $00_{(16)}$ == $0_{(10)}$

```
#include <stdio.h>

int main(void)
{
    char a[4] = { 97, 98, 99, 0 };
    char b[4] = { 'a', 'b', 'c', '\0' };

    return 0;
}
```

문자열(string)

- 문자열의 길이
  - **전달하고자 하는 문자열 내용의 길이**
  - '\0' (NULL)이 나타나기 전까지 문자 코드의 개수
- 문자 배열의 길이
  - 문자열을 저장하는데 사용하는 **배열의 원소 개수**

```
#include <stdio.h>

int main(void)
{
    char a[4] = { 97, 98, 99, 0 };
    char b[4] = { 'a', 'b', 'c', '\0' };

    return 0;
}
```

문자열 a의 길이: 3
(배열 a에 저장되어 있는 문자열의 길이)

배열 a의 길이: 4
(배열 a의 원소의 개수)

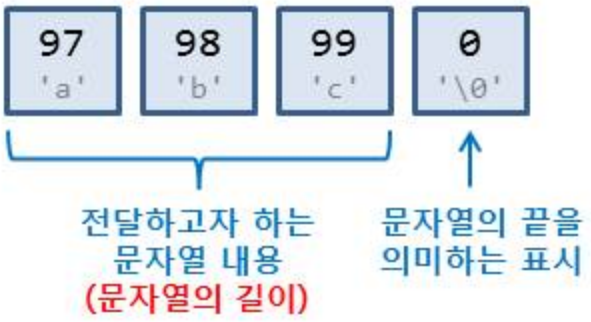
문자열(string)

- 문자열 초기화
  - 배열 초기화 방법
  - **큰 따옴표(")를 사용한 방법**
    - char 타입의 배열에만 가능
    - 마지막 따옴표에 '\0'이 생략되어 있다.

```
#include <stdio.h>

int main(void)
{
    char a[4] = { 97, 98, 99, 0 };
    char b[4] = { 'a', 'b', 'c', '\0' };
    char c[4] = "abc";

    return 0;
}
```

동일한 내용의 초기화

'\0'이 생략되어 있다.



문자열(string)

문자 배열 초기화

```
#include <stdio.h>

int main(void)
{
    char a[4] = {97, 98, 99, 0};
    char b[4] = {'a','b','c','\0'};
    char c[4] = "abc";

    return 0;
}
```

원소 개수의 생략

```
#include <stdio.h>

int main(void)
{
    char a[] = {97, 98, 99, 0};
    char b[] = {'a','b','c','\0'};
    char c[] = "abc";

    return 0;
}
```

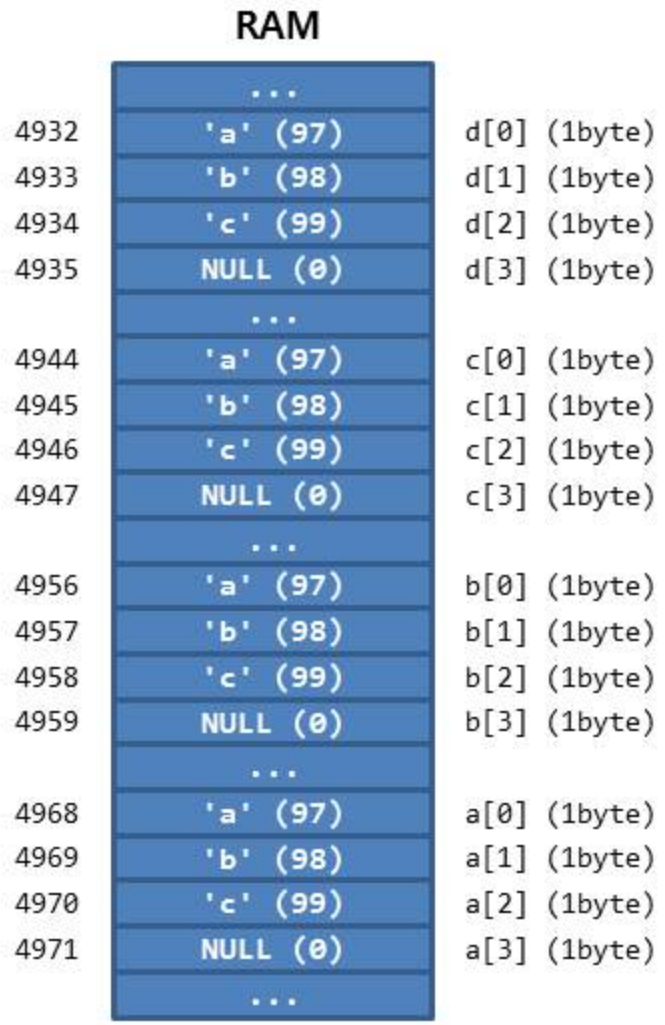
원소의 개수 4로 간주

컴파일러가 배열 intializer 안에
값의 개수를 파악하여
생략된 배열의 원소 개수를 추정함.

문자열(string)

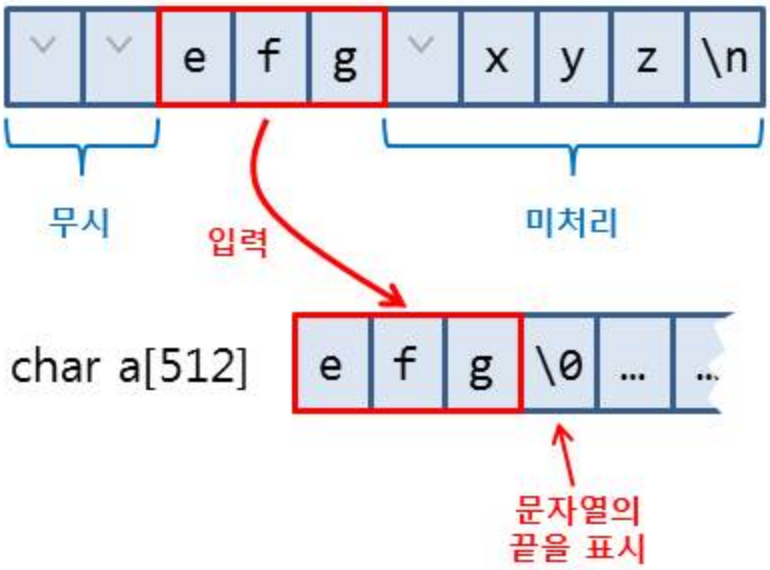
메모리 구조

```c
#include <stdio.h>

int main(void)
{
    char a[4] = { 'a', 'b', 'c', '\0' };
    char b[4] = { 97, 98, 99, 0 };
    char c[4] = "abc";
    char d[] = "abc";

    return 0;
}
```



문자열(string)

- 문자열 입출력을 위한 FSF
  - %s : 문자열 출력
  - scanf 함수를 사용한 입력시 빈칸(space)는 무시되어 입력되지 않는다.




```c
#include <stdio.h>

int main(void)
{
    char a[512];

    scanf("%s",a);
    printf("(%s)\n",a);
    return 0;
}
```

입출력 결과

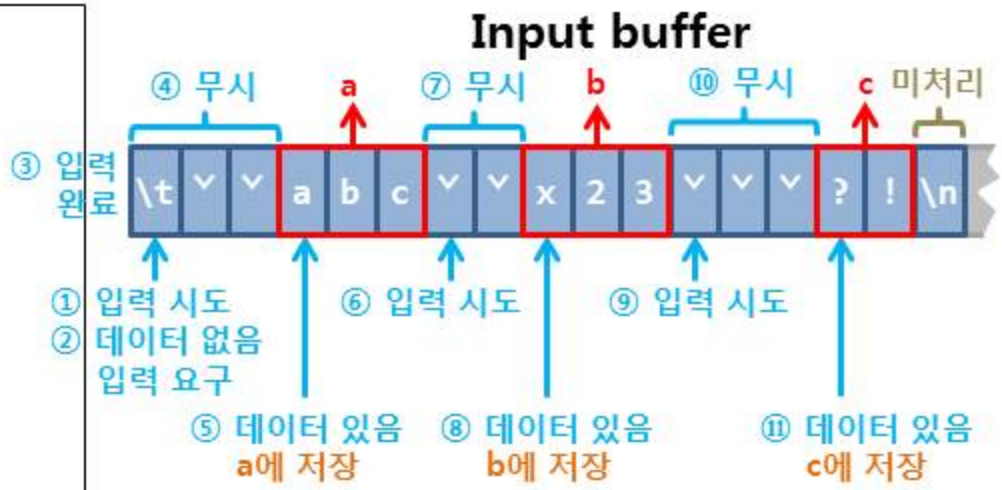
```
abc
(abc)
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

```
  efg xyz
(efg)
계속하려면 아무 키나 누르십시오 . . .
```

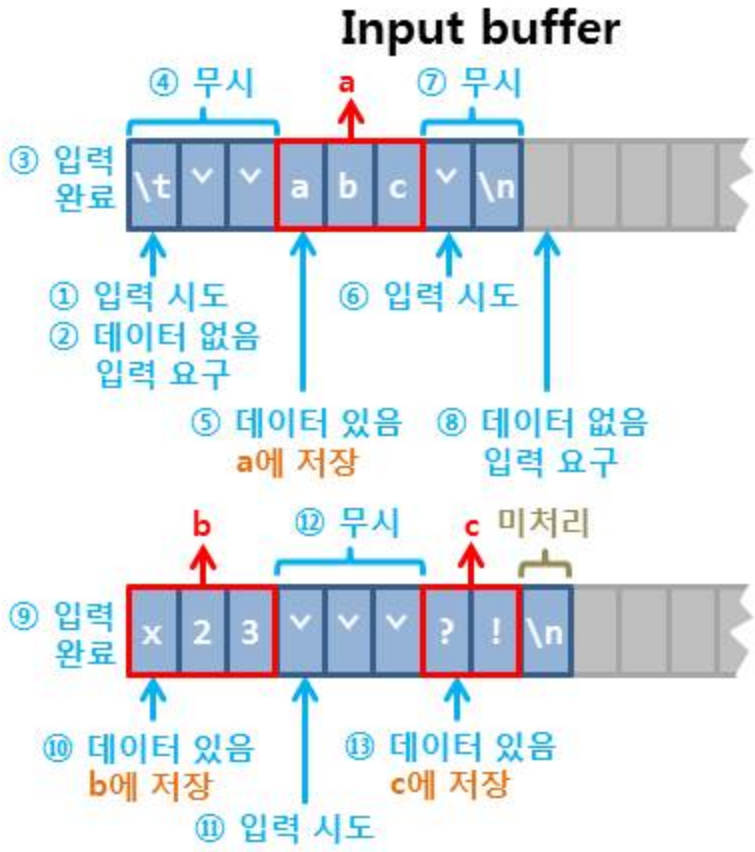
입력 버퍼의 동작 (단어 단위 문자열 입력)

```
#include <stdio.h>

int main(void)
{
    char a[512], b[512], c[512];


                char a[512]  a b c \0

    scanf("%s", a );
    printf("a==%s\n", a );

                char b[512]  x 2 3 \0
                char c[512]  ? ! \0

    scanf("%s%s", b, c );
    printf("b==%s, ", b );
    printf("c==%s\n", c );
    return 0;
}
```



입력 버퍼의 동작 (단어 단위 문자열 입력)

```
#include <stdio.h>

int main(void)
{
    char a[512], b[512], c[512];


                    char a[512]  a b c \0

    scanf("%s", a );
    printf("a==%s\n", a );

                    char b[512]  x 2 3 \0
                    char c[512]  ? ! \0

    scanf("%s%s", b, c );
    printf("b==%s, ", b );
    printf("c==%s\n", c );
    return 0;
}
```



문자열(string)

- %s FSF(Format Specification Field)

Variable i	condition a[i]!='\0'	statement putchar(a[i])

Analysis	Value
The range of i in for-stat.	
The value of i after for-stat.	
The number of iteration	

```
#include <stdio.h>

int main(void)
{
    char a[4] = {'a','b','c','\0'};
    int i;

    for(i=0; a[i]!='\0'; ++i)
        putchar(a[i]);
    putchar('\n');

    printf("%s\n",a);

    return 0;
}
```

동일 결과

문자열의 인자 전달

- 문자열의 길이 전달
  - 문자열의 마지막에는 반드시 '\0'이 저장되어 있다.
  - 문자열의 길이가 간접 전달하면 배열의 길이 전달이 필요 없다.

입출력 결과

```
abcd
abcd
계속하려면 아무 키나 누르십시오 . . .
```

```c
#include <stdio.h>

void f(char s[])
{
    int i;

    for(i=0; s[i]!='\0'; ++i)
        putchar(s[i]);
    putchar('\n');
}

int main(void)
{
    char a[5]={'a', 'b', 'c', 'd', '\0'};
    char b[5]="abcd";

    f(a);
    f(b);
    return 0;
}
```

배열 a의 길이: 5
문자열 a의 길이: 4

배열 b의 길이: 5
문자열 b의 길이: 4

문자열의 길이 vs. char 배열의 길이

배열이 큰 경우

```
#include <stdio.h>

void f(char s[])
{
    int i;

    for(i=0; a[i]!='\0'; ++i)
        putchar(a[i]);
    putchar('\n');
}

int main(void)
{
    char a[10]={'a', 'b', 'c', 'd', '\0'};

    f(a);
    return 0;
}
```

배열 a의 길이: 10
문자열 a의 길이: 4

중간에 NULL이 있는 경우

```
#include <stdio.h>

void f(char s[])
{
    int i;

    for(i=0; a[i]!='\0'; ++i)
        putchar(a[i]);
    putchar('\n');
}

int main(void)
{
    char a[10]="ab\0d";

    f(a);
    return 0;
}
```

배열 a의 길이: 10
문자열 a의 길이: 2

문자열 입출력 함수

■ 입력 함수

```
char* gets(char buffer[]);
```

- 키보드에서 문자열을 입력 받아 문자 배열 buffer에 저장한다.
- 이때 newline(enter key)를 입력 받을 때까지 모든 문자를 buffer에 저장한다. 하지만, newline은 buffer에 저장되지 않는다.
- 배열 buffer의 마지막에는 '\0' 가 저장된다.
- 배열 buffer의 주소가 반환되며, EOF일 때는 0(NULL)이 반환된다.

■ 출력 함수

```
int puts(const char buffer[]);
```

- 전달받은 문자열을 화면에 출력한다. 이때 newline도 자동으로 추가된다.

```
#include <stdio.h>

int main(void)
{
    char buffer[512];

    if( gets(buffer) != NULL ) {
        puts(buffer);  // adding newline
        printf("(%s)\n", buffer);
    }
    return 0;
}
```

입출력 결과

```
abcd 123
abcd 123
(abcd 123)
계속하려면 아무 키나 누르십시오 . . .
```

입출력 결과

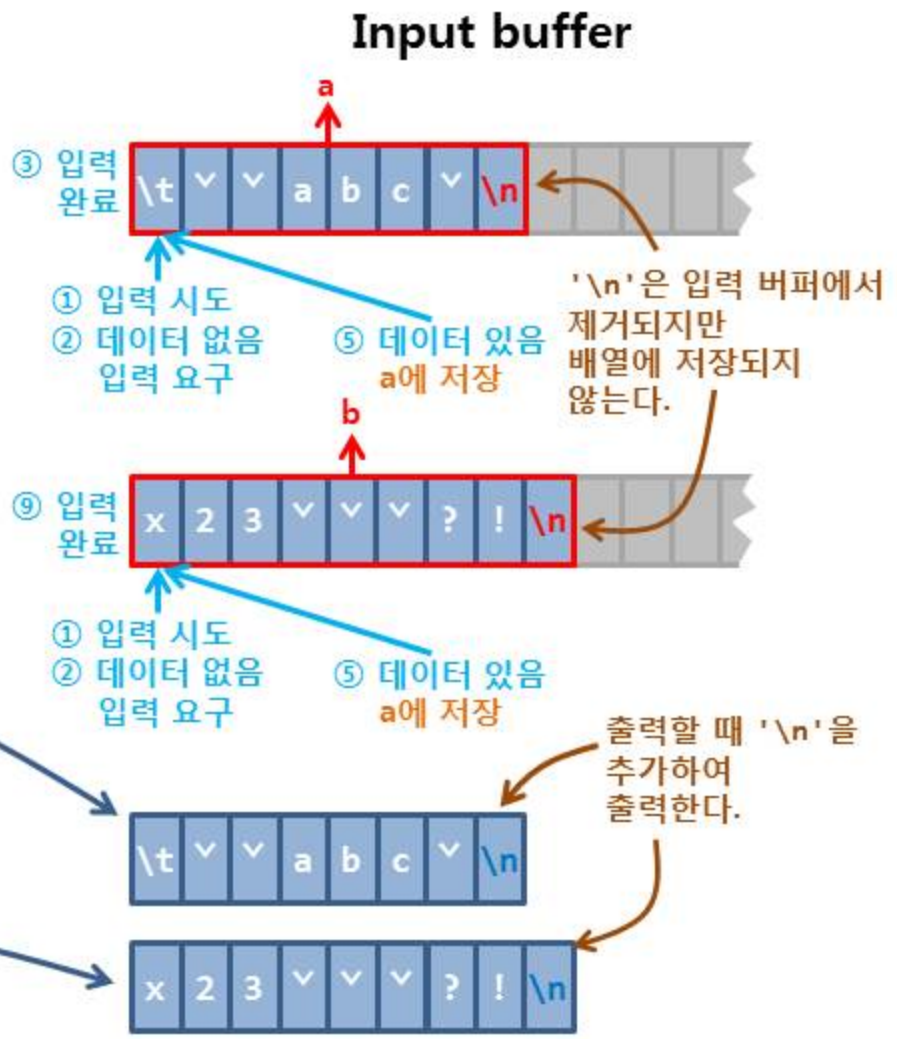
```
^Z
계속하려면 아무 키나 누르십시오 . . .
```

입력 버퍼의 동작 (줄 문자열 단위 입력)

```
#include <stdio.h>

int main(void)
{
    char a[512], b[512];

    gets( a );
    puts( a );

    gets( b );
    puts( b );
    return 0;
}
```

char a[512] \t ˅ ˅ a b c ˅ \0

char b[512] x 2 3 ˅ ˅ ˅ ? ! \0



타입의 재정의

- typedef
  - 기존의 타입을 다른 이름으로 대체할 때 사용

```
typedef <old_type> <new_type>;
```

```
#include <stdio.h>

int main(void)
{
    typedef   int         INT  ;
    typedef   long long   LLONG ;
    typedef   long long int  llong ;

    INT a;
    LLONG b;
    llong c;

    return 0;
}
```