
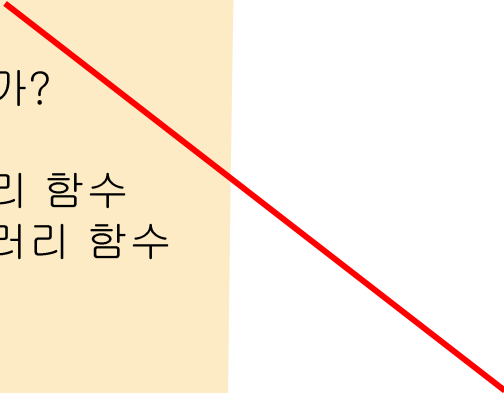


# 제 12장 문자와 문자열

# 이번 장에서 학습할 내용

- 
- 문자 표현 방법
  - 문자열 표현 방법
  - 문자열이란 무엇인가?
  - 문자열의 입출력
  - 문자처리 라이브러리 함수
  - 표준입출력 라이브러리 함수
- 

인간은 문자를 사용하여 정보를 표현하므로 문자열은 프로그램에서 중요한 위치를 차지하고 있다.  
이번 장에서는 C에서의 문자열 처리 방법에 대하여 자세히 살펴볼 것이다.

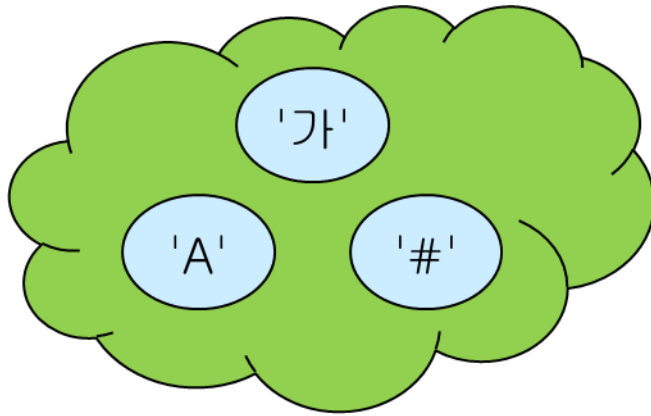


# 문자의 중요성

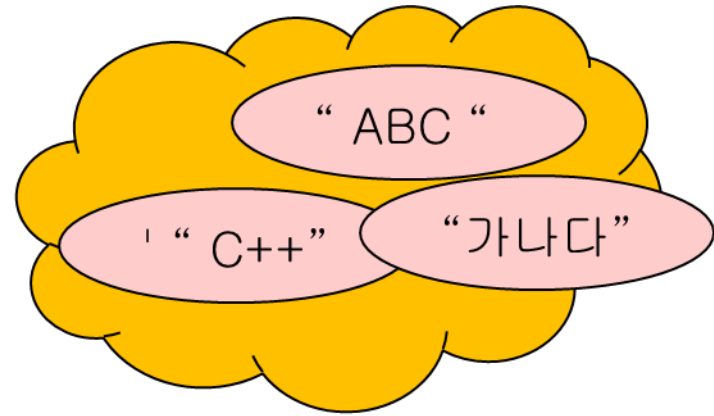
- 인간에게 텍스트는 대단히 중요하다.



# 문자와 문자열



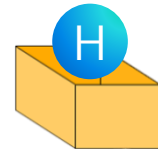
문자



문자열

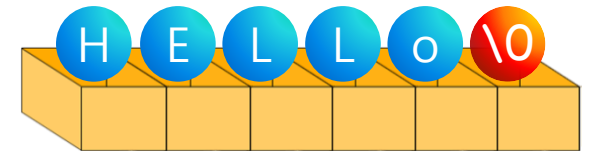
# 문자열 표현 방법

- *문자열(string)*: 문자들이 여러 개 모인 것
  - "A "
  - "Hello World!"
  - "변수 score의 값은 %d입니다"
- *문자열 변수*
  - 변경가능한 문자열을 저장할 수 있는 변수



하나의 문자는 char형 변수로 저장

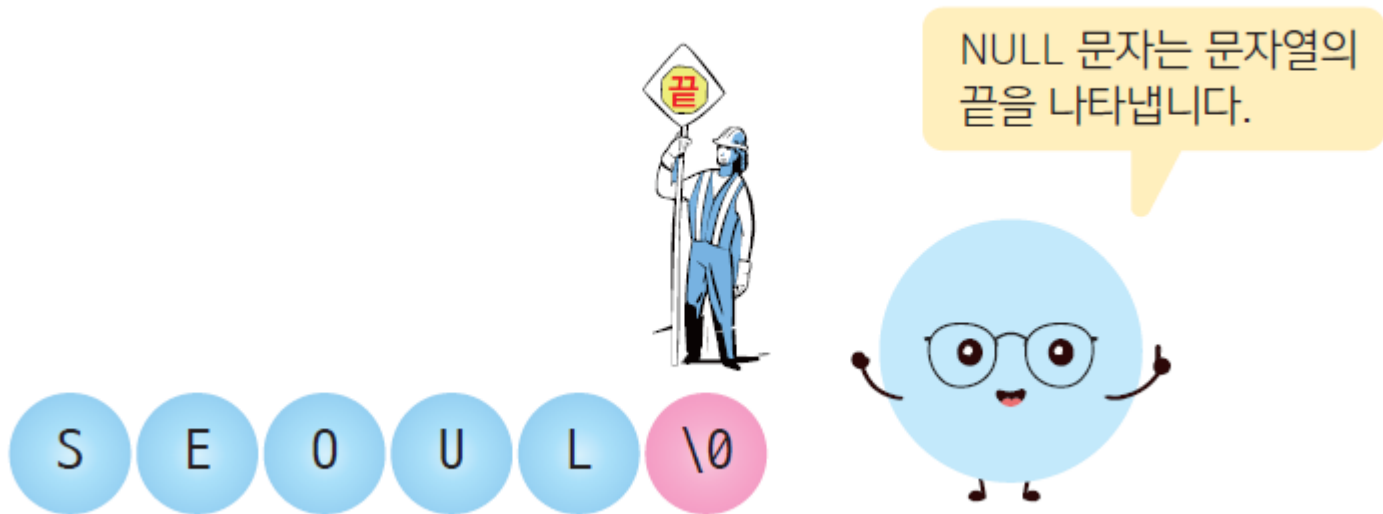
Storage Type	Syntax	Mutable	Auto cleanup	Use when...
Stack	char str[] = "hi";	✓	✓	Short-lived, fast
Static (literal)	char* str = "hi";	✗	✓	Constant strings
Heap	malloc/free	✓	✗	Dynamic/larger strings



문자열은 char형 배열로 저장

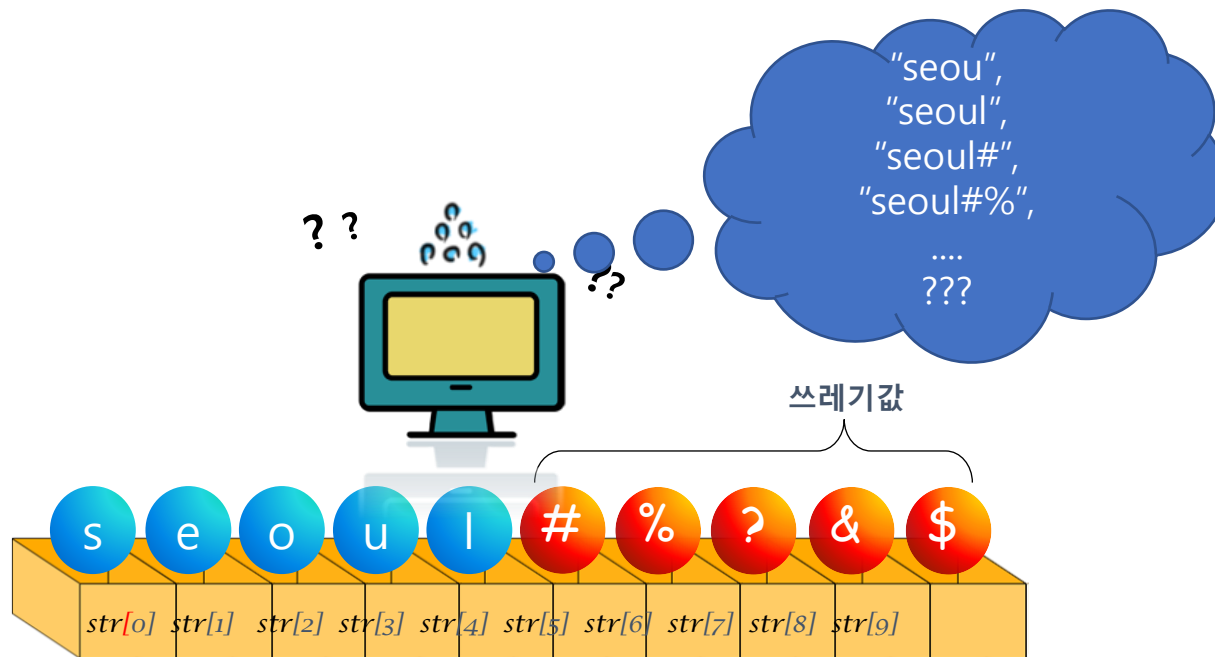
# NULL 문자

- NULL 문자: 문자열의 끝을 나타낸다.



# 왜 문자열의 끝을 표시하여야 하는가?

- 문자열은 어디서 종료되는지 알 수가 없으므로 표시를 해주어야 한다.



# 예제 #1

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int i;
```

```
    char str[4];
```

```
    str[0] = 'a';
```

```
    str[1] = 'b';
```

```
    str[2] = 'c';
```

```
    str[3] = '\0';
```

```
    i = 0;
```

```
    while(str[i] != '\0') {  
        printf("%c", str[i]);
```

```
        i++;
```

```
    }
```

```
    return 0;
```

```
}
```

문자 배열에 들어 있는 문자들을 하나씩 출력하여 보자. 문자 배열에 저장된 문자열을 출력할 때는 %s를 사용하면 되지만 여기서는 문자열 처리의 기본적인 방법을 실감하기 위하여 문자 배열에 들어 있는 문자들을 하나씩 화면에 출력하다가 NULL 문자가 나오면 반복을 종료하도록 하였다.



abc



# 참고

## 참고사항

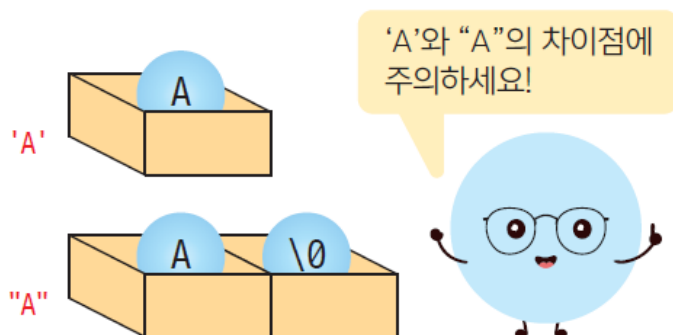
여기서 잠깐 C 언어 퀴즈를 풀고 지나가자. C 언어 코드에서 A, 'A', "A"의 차이를 생각해보자.

A : 컴파일러는 A를 변수의 이름으로 간주한다.

'A' : 문자 A를 나타낸다.

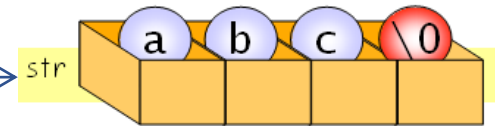
"A" : 문자 A만으로 이루어진 문자열을 나타낸다. 'A'와는 다르다.

여기서 주의해야 할 것은 'A'와 "A"의 차이점이다. 'A'는 하나의 문자를 나타내며 문자 A에 대한 아스키 코드와 같다. "A"는 문자열이며 A의 아스키 코드에 문자열 끝을 나타내는 NULL 문자가 추가된다.

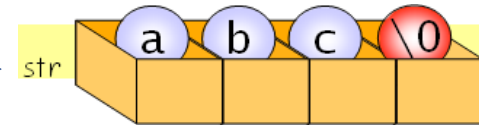


# 문자 배열의 초기화

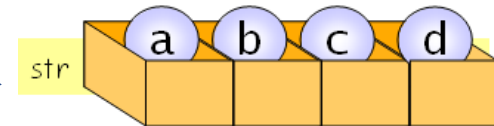
- `char str[4] = { 'a', 'b', 'c', '\0' };`



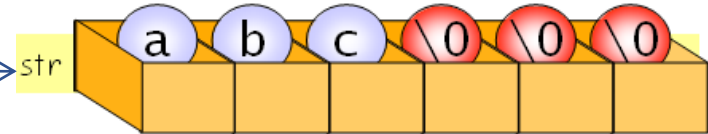
- `char str[4] = "abc";`



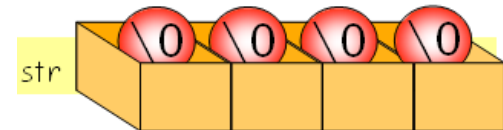
- `char str[4] = "abcdef";`



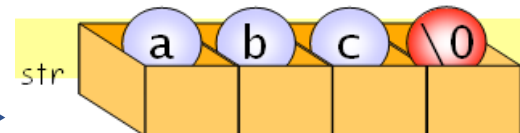
- `char str[6] = "abc";`



- `char str[4] = "";`

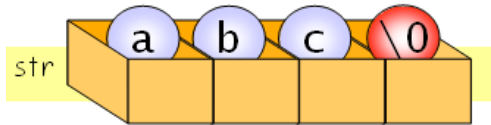


- `char str[] = "abc";`



`char str[];` 오류 → Compiler는 얼마만큼의 메모리를 할당해야 하는지 알수 없다

# 문자열의 출력



```
char str[] = "abc";  
printf("%s", str);
```

```
char str[] = "abc";  
printf(str);
```



## 예제 #2

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    char str1[6] = "Seoul";
```

```
    char str2[3] = { 'i', 's', '\0' };
```

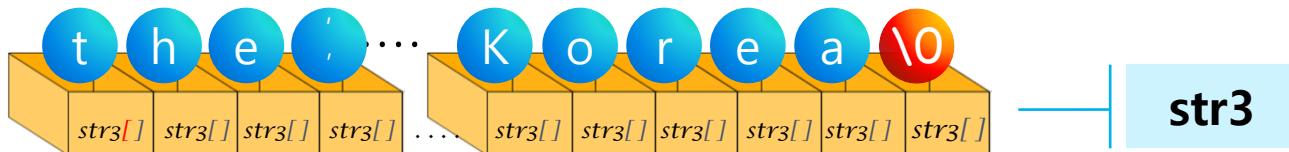
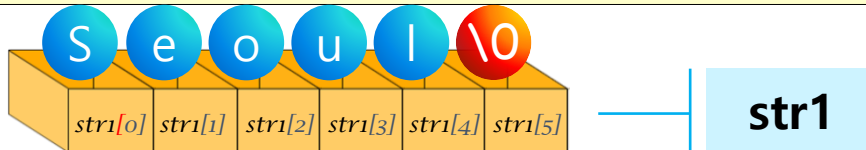
```
    char str3[] = "the capital city of Korea.";
```

```
    printf("%s %s %s\n", str1, str2, str3);
```

```
    return 0;
```

```
}
```

Seoul is the capital city of Korea.



## 예제 #3 (string copy)

```
#include <stdio.h>

int main(void)
{
    char src[] = "Action speaks louder than words";
    char dst[100];

    int i;
    printf("원본 문자열=%s\n", src);
    for (i = 0; src[i] != '\0'; i++)
        dst[i] = src[i];
    dst[i] = '\0';
    printf("복사된 문자열=%s\n", dst);

    return 0;
}
```

NULL과 '\0'은 같다.

원본 문자열=Action speaks louder than words  
복사된 문자열=Action speaks louder than words

# 문자열 길이 계산 예제

```
// 문자열의 길이를 구하는 프로그램
#include <stdio.h>

int main(void)
{
    char str[30] = "C language is easy";
    int i = 0;

    while(str[i] != 0)
        i++;

    printf("문자열 \"%s\"의 길이는 %d입니다.\n", str, i);
    return 0;
}
```

문자열 "C language is easy"의 길이는 18입니다.

# 문자 배열을 변경하는 방법

1. 첫 번째 방법은 각 배열 요소에 원하는 문자를 개별적으로 대입하는 방법이다. 확실한 방법이지만 아주 불편하다.

```
char str[10]="Hello";  
str[0] = 'W';  
str[1] = 'o';  
str[2] = 'r';  
str[3] = 'l';  
str[4] = 'd';  
str[5] = '\0';
```

2. 라이브러리 함수인 strcpy()를 사용하여 문자열을 문자 배열에 복사할 수 있다.

```
char str[10]="Hello";  
strcpy(str, "World");
```

# 잘못된 방법

3. 가장 편리할 것 같은, 다음과 같은 방법은 사용할 수 없다. 주의하여야 한다.

```
char str[10] = "Hello";  
str = "World";           // 문법적인 오류!
```

배열에서 학습하였지만  
배열의 이름은 배열을 가  
리키는 포인터 상수입니  
다. 변경이 불가능합니다.





# 문자열 상수

- 문자열 상수: "HelloWorld"와 같이 프로그램 소스 안에 포함된 문자열
- 문자열 상수는 메모리 영역 중에서 **텍스트 세그먼트(text segment)** 에 저장

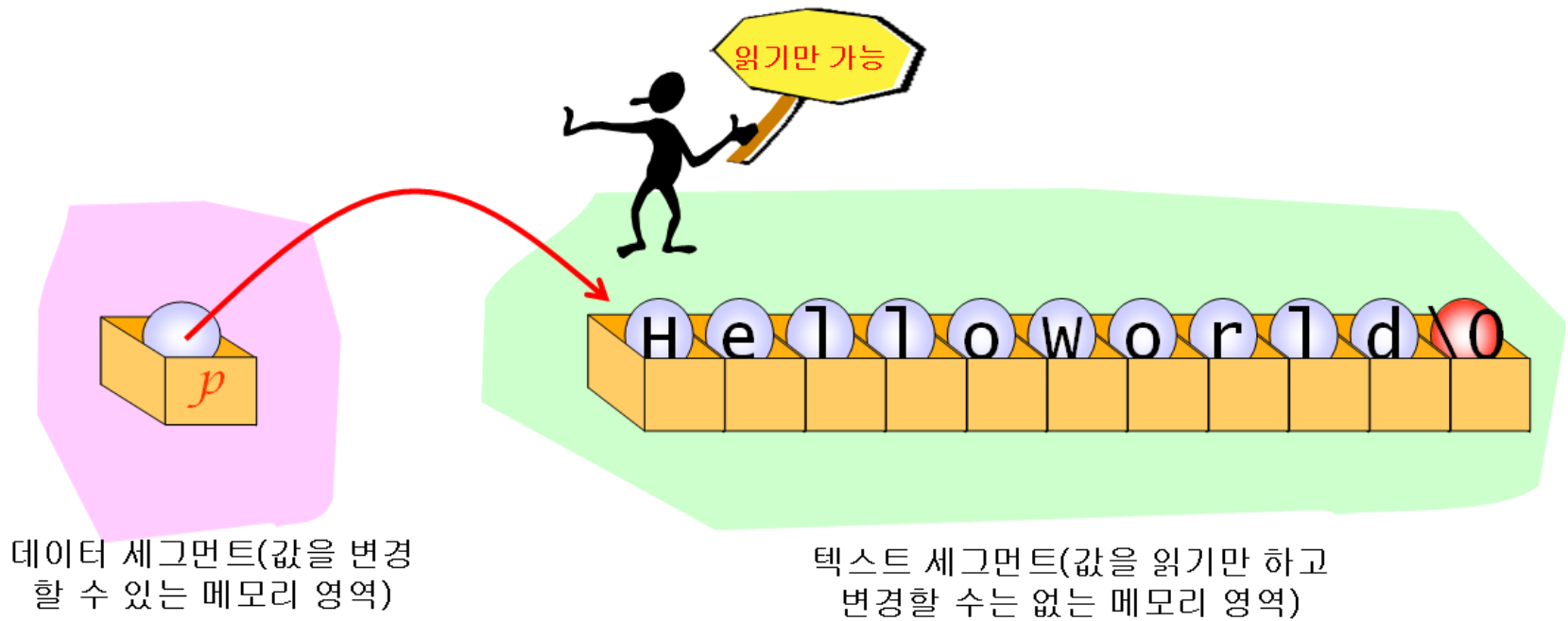
```
char *p = "HelloWorld";
```

위 문장의 정확한  
의미는 무엇일까요?



# 문자열 상수

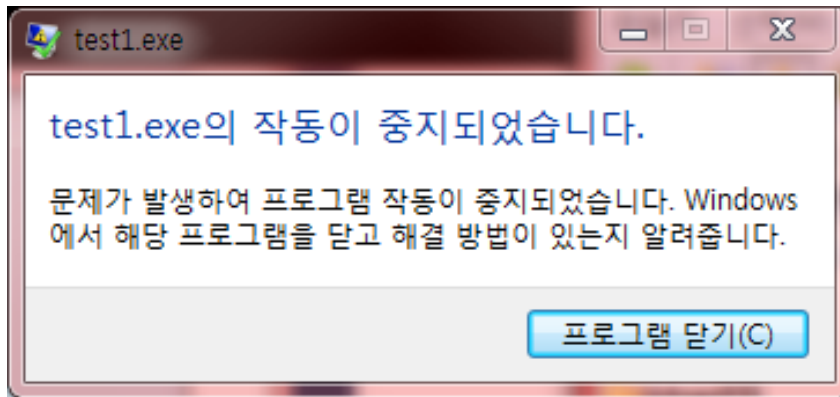
```
char *p = "HelloWorld";
```



# 문자열 상수

```
char *p = "HelloWorld";  
strcpy(p, "Goodbye");
```

p를 통하여 텍스트 세그먼트에 문자를  
저장하려면 오류가 발생한다.

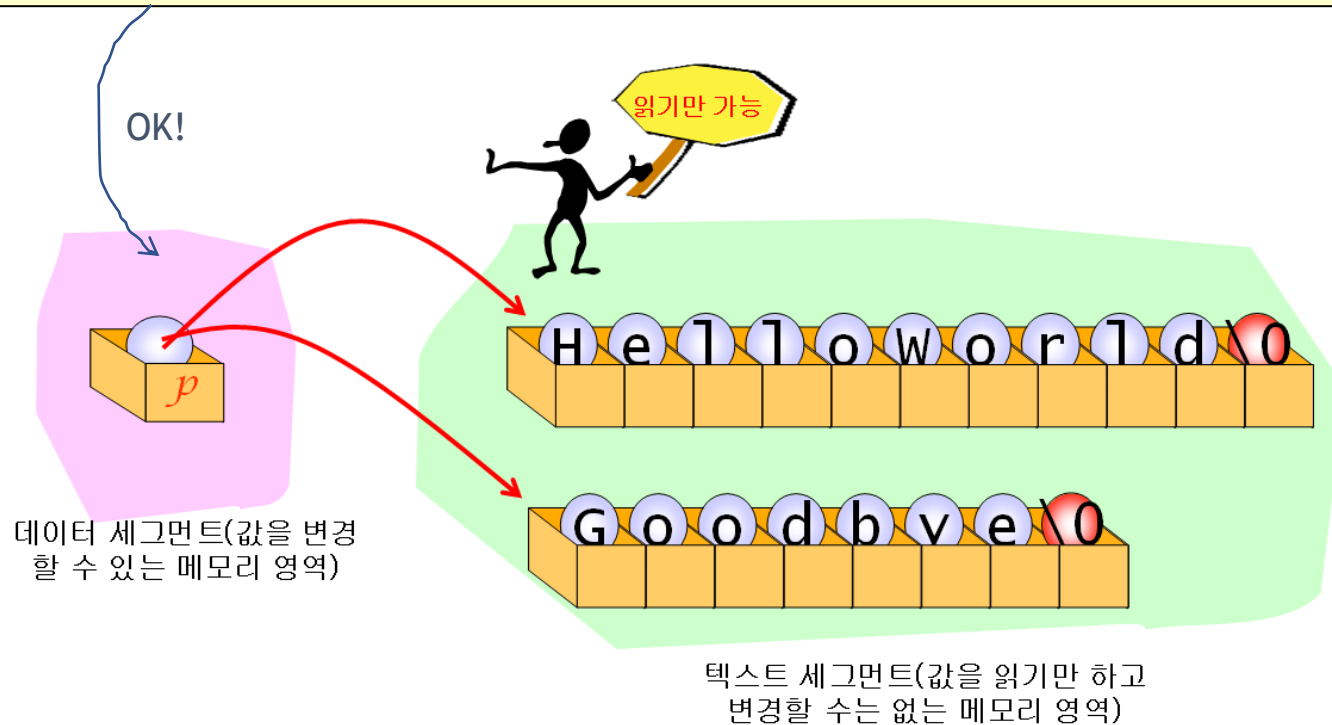


텍스트 세그먼트는  
변경할 수 없습니다.



# 문자열 상수

```
char *p = "HelloWorld";  
p = "Goodbye";
```



# 예제

```
#include <stdio.h>

int main(void)
{
    char* p = "HelloWorld";
    printf("%s \n", p);

    p = "Welcome to C World!"; // 가능
    printf("%s \n", p);

    p = "Goodbye"; // 가능
    printf("%s \n", p);
    // p[0] = 'a'; // 오류가 발생한다.

    return 0;
}
```

```
HelloWorld
Welcome to C World!
Goodbye
```

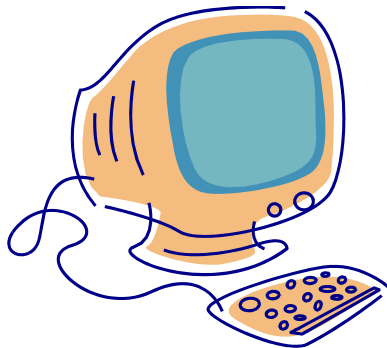
# 중간 점검

1. C에서 문자열은 어떻게 정의되는가?
2. 문자열에서 NULL 문자의 역할은 무엇인가?
3. NULL 문자의 아스키 코드 값은 얼마인가?
4. NULL 문자로 끝나지 않는 문자열을 출력하면 어떻게 되는가?
5. 'B', "B"의 차이점을 설명하라.
6. 변경 가능한 문자열은 어디에 저장되는가?
7. 문자열의 크기보다 문자 배열의 크기를 하나 더 크게 하는 이유는 무엇인가?
8. 문자 배열을 문자열로 초기화하는 방법을 아는 대로 설명하라.

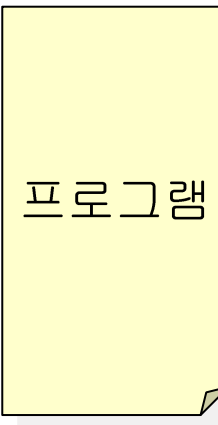


# 문자 입출력 라이브러리

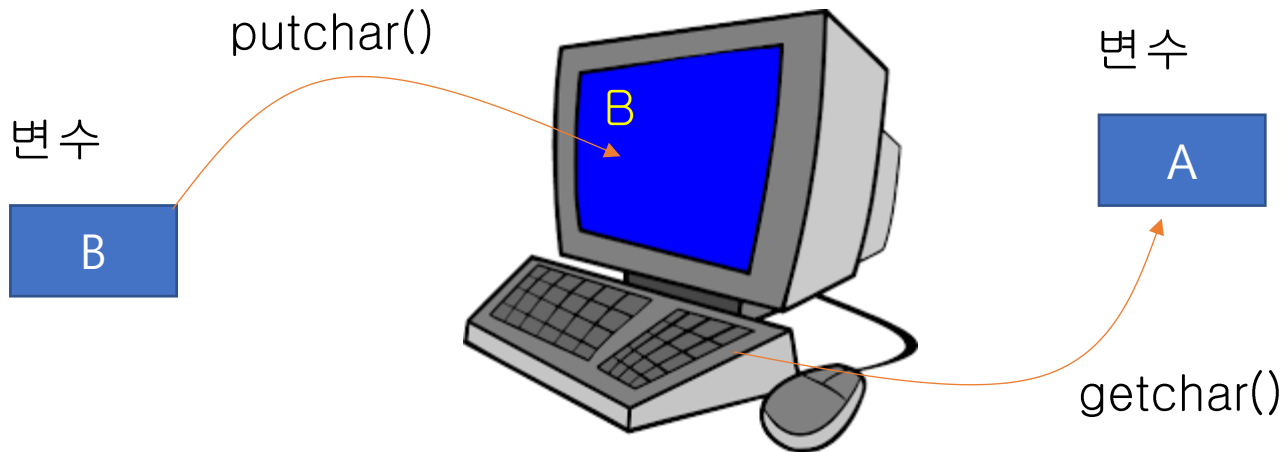
입출력 함수	설명
<code>int getchar(void)</code>	하나의 문자를 읽어서 반환한다.
<code>void putchar(int c)</code>	변수 <code>c</code> 에 저장된 문자를 출력한다.
<code>int _getch(void)</code>	하나의 문자를 읽어서 반환한다(버퍼를 사용하지 않음).
<code>void _putch(int c)</code>	변수 <code>c</code> 에 저장된 문자를 출력한다(버퍼를 사용하지 않음).
<code>scanf("%c", &amp;c)</code>	하나의 문자를 읽어서 변수 <code>c</code> 에 저장한다.
<code>printf("%c", c);</code>	변수 <code>c</code> 에 저장된 문자를 출력한다.



...'A' 'B' 'C' ...



# getchar(), putchar()





# getchar(), putchar()

```
// getchar()의 사용
```

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int ch;    // 정수형에 주의
```

```
    while( (ch = getchar()) != EOF)
```

```
        putchar(ch);
```

```
    return 0;
```

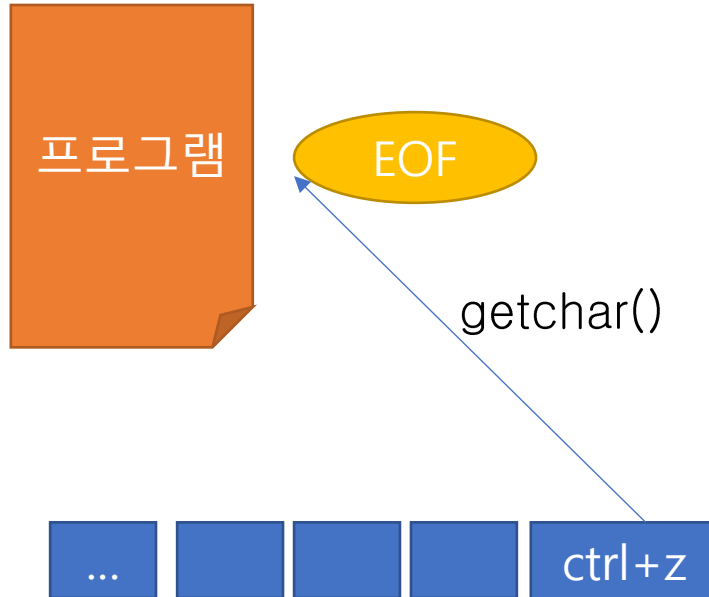
```
}
```

End Of File을  
나타내는 문자,  
EOF는 정수형  
-1이다.

```
a  
a  
b  
b  
^Z
```

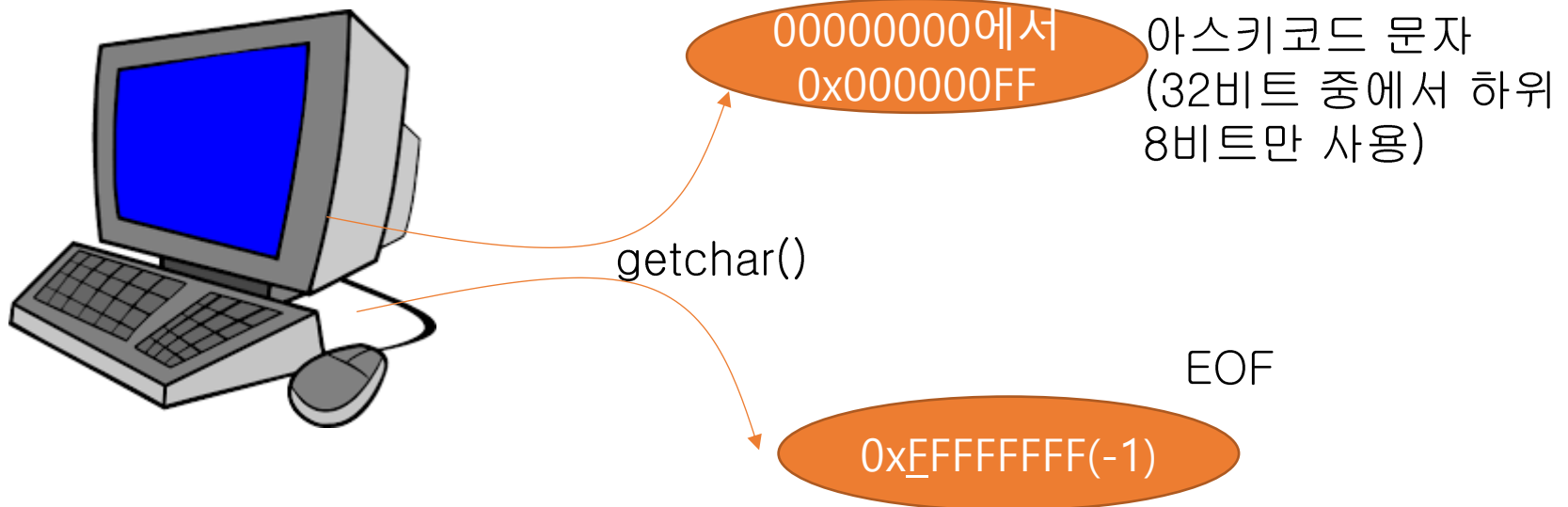
# EOF

- EOF(End Of File): 입력의 끝을 나타내는 특수한 기호



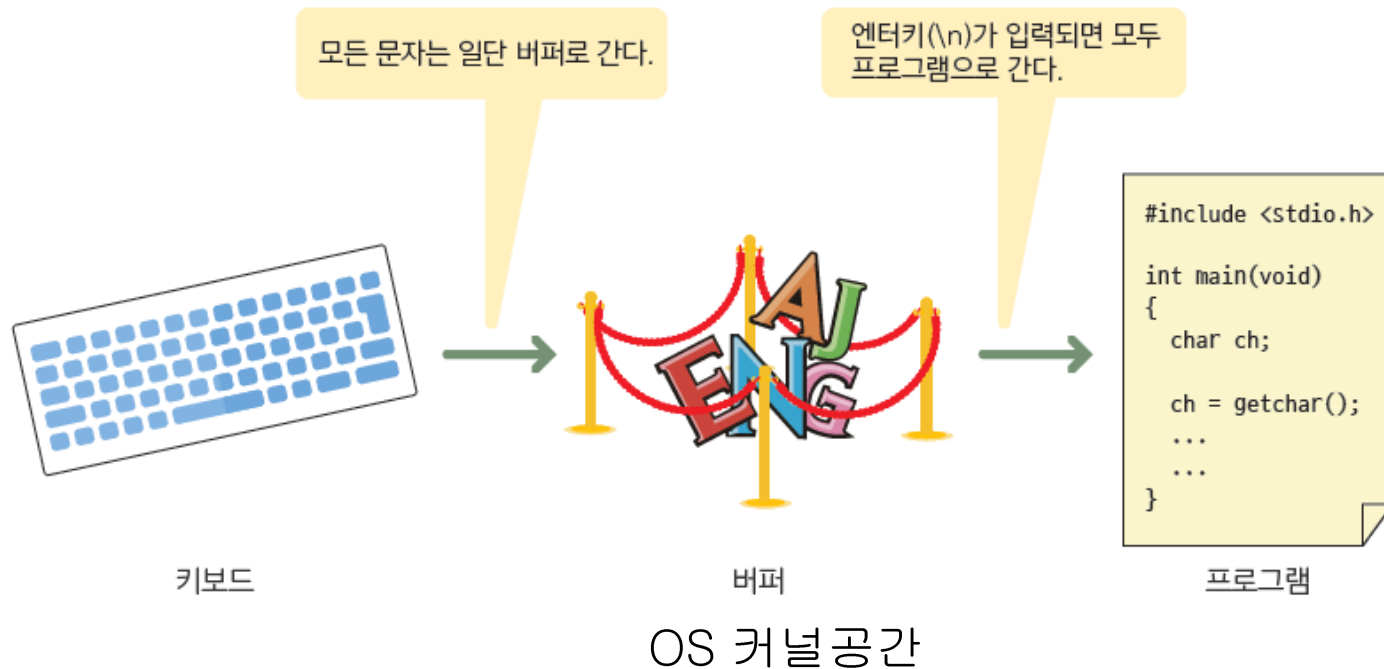
# getchar()의 반환값이 정수형인 이유

- getchar( )의 반환형이 int형인 것은 아스키 코드를 EOF와 구별하기 위해서이다.
- 반환값을 32비트 int형으로 하면 아스키코드 (0x00000000에서 0x000000FF)와 32비트 int 형-1인 EOF(0xFFFFFFFF)와 확실하게 구분할 수 있다. 반환값을 8비트를 하면 불가능하다.



# 버퍼링

- 엔터키를 쳐야만 입력을 받는 이유



# \_getch(), \_getche(), getchar()

	헤더파일	버퍼사용여부	에코여부	응답성	문자수정여부
getchar()	<stdio.h>	사용함 (엔터키를 눌러입력됨)	에코	줄단위	가능
_getch()	<conio.h>	사용하지 않음	에코하지 않음	문자단위	불가능
_getche()	<conio.h>	사용하지 않음	에코	문자단위	불가능



용도에 맞는 것을  
골라 사용하세요!

버퍼가 없이 바로  
받으려면  
**\_getch()**를  
사용합니다.

# \_getch(), \_putch()

Edit 편집기 프로그램에 유용!

```
#include <stdio.h>
#include <conio.h>
```

```
int main(void)
```

```
{
```

```
    int ch;
```

```
    while( (ch = _getch()) != 'q' )
```

```
        _putch(ch);
```

```
    return 0;
```

```
}
```

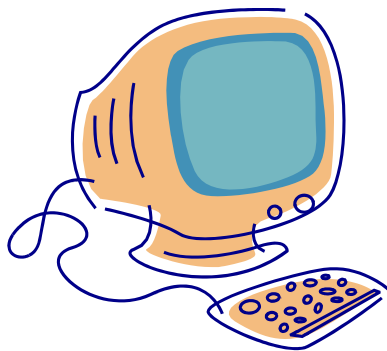
버퍼를 사용하지  
않는다,  
에코도 없음!

abc

# 문자열 입출력 라이브러리 함수

Windows

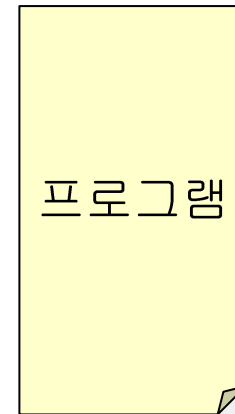
입출력 함수	설명
<code>int scanf("%s", s)</code>	문자열을 읽어서 문자배열 <code>s[]</code> 에 저장
<code>int printf("%s", s)</code>	배열 <code>s[]</code> 에 저장되어 있는 문자열을 출력한다.
<code>char *gets_s(char *s, int size)</code>	한 줄의 문자열을 읽어서 문자 배열 <code>s[]</code> 에 저장한다.
<code>int puts(const char *s)</code>	배열 <code>s[]</code> 에 저장되어 있는 한 줄의 문자열을 출력한다.



...Hello World!...



프로그램



# 예제

```
#include <stdio.h>

int main(void)
{
    char name[100];
    char address[100];

    printf("이름을 입력하시오: ");
    scanf("%s", name);
    printf("현재 거주하는 주소를 입력하시오: ");
    scanf("%s", address);

    printf("안녕하세요, %s에 사는 %s씨.\n", address, name);
    return 0;
}
```

주소가 다 저장되지 못한다.

이름을 입력하시오: 홍길동  
현재 거주하는 주소를 입력하시오: 서울시 종로구 1번지  
안녕하세요. 서울시에 사는 홍길동씨.



# gets\_s()와 puts() 문자열 입출력

Syntax

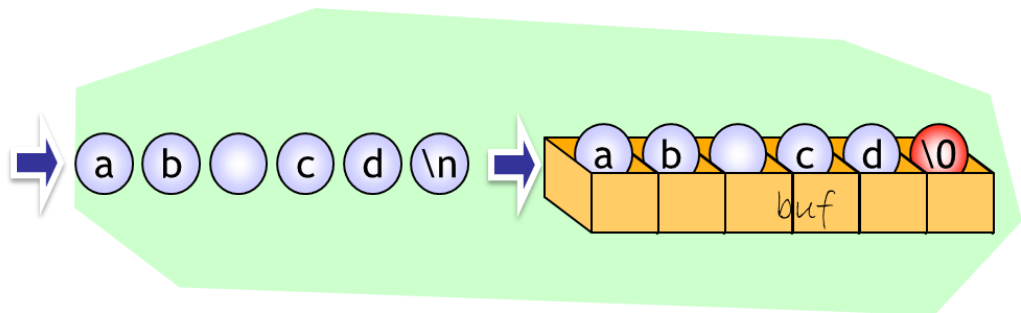
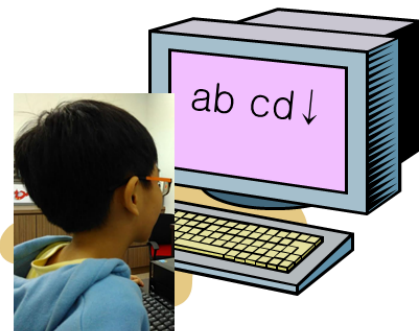
gets\_s()

예

```
char buf[100];  
gets_s(buf, 100);  
puts(buf);
```

사용자로부터 한 줄을 입력받는다.

한 줄을 출력한다.



gets\_s(buf, 100);

# 예제

```
#include <stdio.h>
int main(void)
{
    char name[100];
    char address[100];

    printf("이름을 입력하시오: ");
    gets_s(name, sizeof(name));
    printf("현재 거주하는 주소를 입력하시오: ");
    gets_s(address, sizeof(address));

    printf("안녕하세요, %s에 사는 %s씨.\n", address, name);
    return 0;
}
```

한 단어 이상을 입력  
받을 때에 사용한다.

이름을 입력하시오: 홍길동  
현재 거주하는 주소를 입력하시오: 서울시 종로구 1번지  
안녕하세요? 서울시 종로구 1번지에 사는 홍길동씨.

# 문자 처리 라이브러리 함수

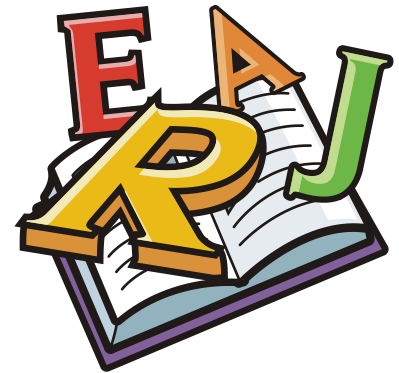
- 문자를 검사하거나 문자를 변환한다.

함수	설명
isalpha(c)	c가 영문자인가?(a-z, A-Z)
isupper(c)	c가 대문자인가?(A-Z)
islower(c)	c가 소문자인가?(a-z)
isdigit(c)	c가 숫자인가?(0-9)
isalnum(c)	c가 영문자이나 숫자인가?(a-z, A-Z, 0-9)
isxdigit(c)	c가 16진수의 숫자인가?(0-9, A-F, a-f)
isspace(c)	c가 공백문자인가?(' ', '\n', '\t', '\v', '\r')
ispunct(c)	c가 구두점 문자인가?
isprint(c)	C가 출력가능한 문자인가?
iscntrl(c)	c가 제어 문자인가?
isascii(c)	c가 아스키 코드인가?

# 문자 처리 라이브러리 함수

- 문자를 검사하거나 문자를 변환한다.

함수	설명
<code>toupper(c)</code>	c를 대문자로 바꾼다.
<code>tolower(c)</code>	c를 소문자로 바꾼다.
<code>toascii(c)</code>	c를 아스키 코드로 바꾼다.



# 예제

```
#include <stdio.h>
#include <ctype.h>
```

```
int main( void )
```

```
{
```

```
    int c;
```

```
    while((c = getchar()) != EOF)
```

```
    {
```

```
        if( islower(c) )
```

```
            c = toupper(c);
```

```
        putchar(c);
```

```
    }
```

```
    return 0;
```

```
}
```

소문자인지 검사

대문자로 변환

```
abcdef
ABCDEF
^Z
```

EOF를 키보드에서 입력하려면 ^Z

# Lab: 단어 세기

- 문자열 안에 들어 있는 단어의 개수를 세는 프로그램을 작성하여 보자. 문자열이 "the c book..."이라면 다음과 같은 출력이 나온다.



단어의 개수: 3

# 예제

```
#include <stdio.h>
#include <ctype.h>

int count_word(char* s);
int main(void)
{
    int wc = count_word("the c book...");
    printf("단어의 개수: %d \n", wc);

    return 0;
}
```

# 예제

```
int count_word(char* s)
{
    int i, wc = 0, waiting = 1;

    for (i = 0; s[i] != NULL; ++i) // s의 각 글자 조사
        if (isalpha(s[i])) // s의 글자가 알파벳이면
        {
            if (waiting) // 단어를 기다리고 있으면
            {
                wc++; // 카운터를 증가
                waiting = 0; // 단어를 처리하는 중
            }
        }
        else // 알파벳이 아니면
            waiting = 1; // 단어를 기다린다.

    return wc;
}
```



# Lab: 유효한 암호 확인

- 유효한 암호는 7 글자 이상이고, 하나 이상의 소문자, 하나 이상의 대문자 및 하나 이상의 숫자를 포함해야 한다.

패스워드를 입력하시오: abc1234

유효한 암호가 아닙니다.

# Solutionn

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(void)
{
    int lower_case_count = 0;        // 소문자 개수
    int upper_case_count = 0;        // 대문자 개수
    int digit_count = 0;             // 숫자 개수
    char pass[100];
    int len;

    printf("패스워드를 입력하시오: ");
    gets_s(pass, sizeof(pass));
```

# Solution

```
len = strlen(pass);      // 문자열의 길이
if (len < 7) {
    printf("유효한 암호가 아닙니다. \n");
    exit(1);
}
for (int i = 0; i < len; i++) {
    if (islower(pass[i])) ++lower_case_count;
    if (isupper(pass[i])) ++upper_case_count;
    if (isdigit(pass[i])) ++digit_count;
}
if (lower_case_count > 0 && upper_case_count > 0 && digit_count > 0)
    printf("강한 암호입니다. \n");
else
    printf("유효한 암호가 아닙니다\n");
return 0;
}
```

# 중간 점검

1. 문자 처리 라이브러리 함수를 사용하려면 포함시켜야 하는 헤더 파일은 무엇인가?
2. getchar()와 getch()가 다른 점은 무엇인가?
3. ispunct('.')의 반환값은 무엇인가?
4. toupper('a')의 반환값은 무엇인가?



# 문자열 처리 라이브러리

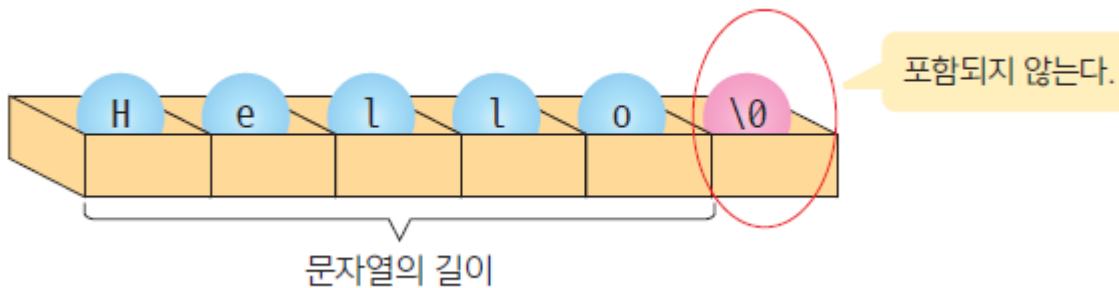
- 문자열 함수들은 `string.h`에 선언되어 있다. 따라서 이들 함수들을 사용하려면 `string.h`를 프로그램의 첫 부분에서 포함시켜야 한다.

함수	설명
<code>strlen(s)</code>	문자열 <code>s</code> 의 길이를 구한다.
<code>strcpy(s1, s2)</code>	<code>s2</code> 를 <code>s1</code> 에 복사한다.
<code>strcat(s1, s2)</code>	<code>s2</code> 를 <code>s1</code> 의 끝에 붙여넣는다.
<code>strcmp(s1, s2)</code>	<code>s1</code> 과 <code>s2</code> 를 비교한다.
<code>strncpy(s1, s2, n)</code>	<code>s2</code> 의 최대 <code>n</code> 개의 문자를 <code>s1</code> 에 복사한다.
<code>strncat(s1, s2, n)</code>	<code>s2</code> 의 최대 <code>n</code> 개의 문자를 <code>s1</code> 의 끝에 붙여넣는다.
<code>strncmp(s1, s2, n)</code>	최대 <code>n</code> 개의 문자까지 <code>s1</code> 과 <code>s2</code> 를 비교한다.
<code>strchr(s, c)</code>	문자열 <code>s</code> 안에서 문자 <code>c</code> 를 찾는다.
<code>strstr(s1, s2)</code>	문자열 <code>s1</code> 에서 문자열 <code>s2</code> 를 찾는다.

# 문자열의 길이

- 문자열의 길이를 계산하는 함수는 `strlen(const char *str)`이다.  
`int size = strlen("Hello");` // size는 5가 된다.

```
char a[6] = "Hello";  
int size = strlen(a);    // size는 5가 된다.
```



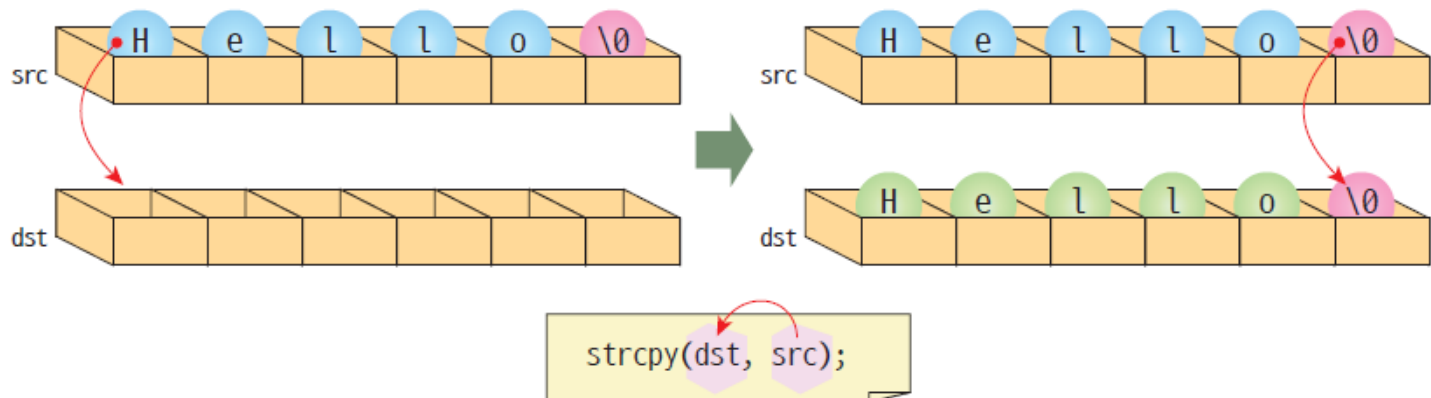
# 문자열 복사

**Syntax** strcpy()

예

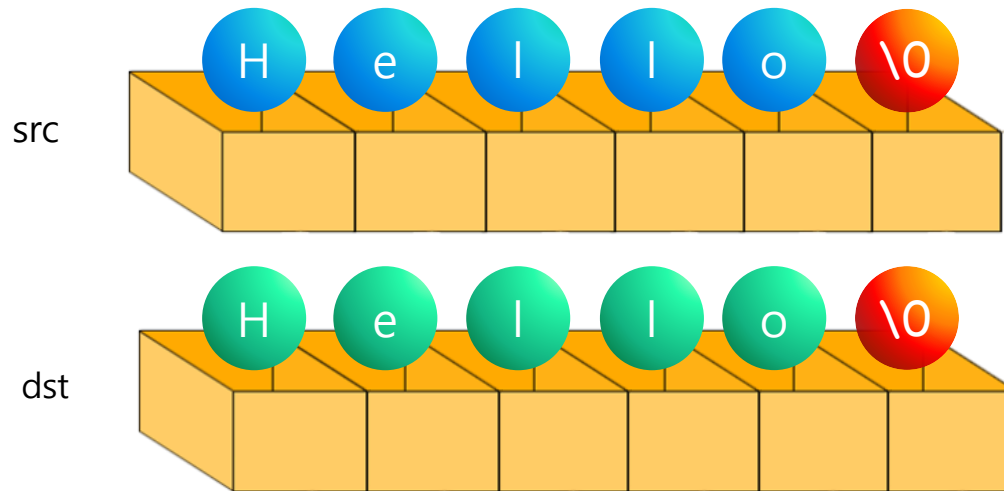
```
char dst[6];  
char src[6]="Hello";  
strcpy(dst, src);
```

// src를 dst로 복사한다.



# 문자열 복사 애니메이션

```
char dst[6];  
char src[6] = "Hello";  
strcpy(dst, src);
```



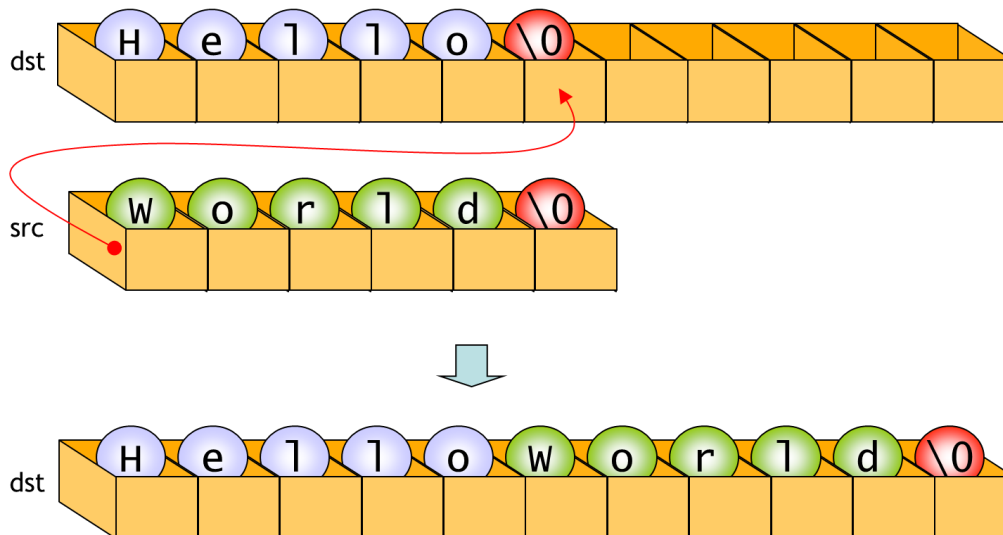


# 문자열 연결

Syntax `strcat()`

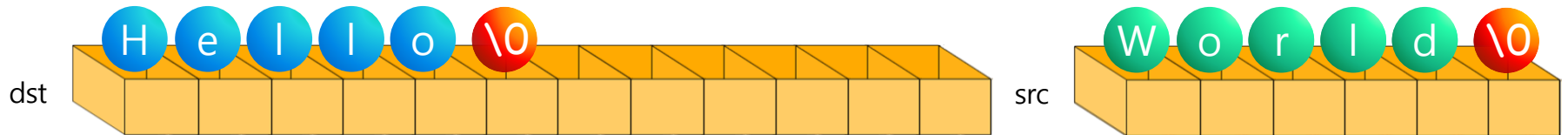
예

```
char dst[12]= "Hello";  
char src[6] = "World";  
strcat(dst, src);           // dst가 "HelloWorld"가 된다.
```



# 문자열 연결 애니메이션

```
char dst[12] = "Hello";  
char src[6] = "World";  
strcat(dst, src);
```



# 예제

```
// strcpy와 strcat
#include <string.h>
#include <stdio.h>

int main( void )
{
    char string[80];

    strcpy( string, "Hello world from " );
    strcat( string, "strcpy " );
    strcat( string, "and " );
    strcat( string, "strcat!" );
    printf( "string = %s\n", string );
    return 0;
}
```

string = Hello world from strcpy and strcat!

# 문자열 비교

## Syntax `strcmp()`

`strcmp()`는 문자열 `s1`과 `s2`를 비교하여 사전적인 (lexicographic) 순서에서 `s1`이 앞에 있으면 음수가 반환되고, 같으면 0이, 뒤에 있으면 양수가 반환된다.

예

```
int result = strcmp("dog", "dog"); // 0이 반환된다.
```

반환값	s1과 s2의 관계
< 0	s1이 s2보다 앞에 있다.
0	s1 == s2
> 0	s1이 s2보다 뒤에 있다.

문자열이 같으면  
`strcmp()`는 0을 반환합  
니다. 주의하세요!



# 예제

```
// strcmp() 함수
#include <string.h>
#include <stdio.h>

int main( void )
{
    char s1[80];    // 첫번째 단어를 저장할 문자배열
    char s2[80];    // 두번째 단어를 저장할 문자배열
    int result;

    printf("첫번째 단어를 입력하시오:");
    scanf("%s", s1);
    printf("두번째 단어를 입력하시오:");
    scanf("%s", s2);
```

# 예제

```
result = strcmp(s1, s2);  
if( result < 0 )  
    printf("%s가 %s보다 앞에 있습니다.\n", s1, s2);  
else if( result == 0 )  
    printf("%s가 %s와 같습니다.\n", s1, s2);  
else  
    printf("%s가 %s보다 뒤에 있습니다.\n", s1, s2);  
return 0;  
}
```

알파벳 순으로 즉 사전에서  
앞에 있다는 의미

첫번째 단어를 입력하시오: cat  
두번째 단어를 입력하시오: dog  
cat가 dog보다 앞에 있습니다.

# 문자 검색

- 주어진 문자열에 특정한 문자가 있는지를 검색하려면 strchr( )를 이용한다.

Syntax

strchr()

예

```
char *p = strchr("dog", 'g');
```

'g' 문자의 주소를 반환한다.

# 문자 검색

```
#include <string.h>
```

```
#include <stdio.h>
```

```
int main( void )
```

```
{
```

```
    char s[] = "language";
```

```
    char c = 'g';
```

```
    char *p;
```

```
    int loc;
```

```
    p = strchr(s, c);
```

```
    if ( p == NULL )
```

```
        printf( "%c가 발견되지 않았음\n", c );
```

```
    else {
```

```
        loc = (int)(p - s);
```

```
        printf("%s에서 첫 번째 %c가 %d에서 발견되었음\n", s, c, loc);
```

```
    }
```

```
    return 0;
```

```
}
```

s 안에서 문자 c를 찾는다.

language에서 첫 번째 g가 3에서 발견되었음



# 문자열 검색

- 주어진 문자열 안에 특정한 문자열이 있는지를 검색하려면 strstr()를 이용한다.

## Syntax

strstr()

strstr() 함수는 문자열 s안에서 부분 문자열 (substring) sub를 검색하는 함수이다. 만약 부분 문자열이 발견되면 그 위치의 주소를 반환한다. 만약 부분 문자열을 찾지 못하면 NULL값이 반환된다.

예

```
char *p = strstr("dog and cat", "cat");
```

# 문자열 검색

codespace

```
#include <string.h>
#include <stdio.h>

int main( void )
{
    char s[] = "A bird in hand is worth two in the bush";
    char sub[] = "bird";
    char *p;
    int loc;

    p = strstr(s, sub);
    if ( p == NULL )
        printf( "%s가 발견되지 않았음\n", sub );
    else {
        loc = (int)(p - s);
        printf( "%s에서 첫 번째 %s가 %d에서 발견되었음\n", s, sub, loc );
    }
    return 0;
}
```

s 안에서 문자열 sub를 찾는다.

A bird in hand is worth two in the bush에서 첫 번째 bird가 2에서 발견되었음

# 문자열 토큰 분리

- strtok() 함수를 사용하면 문장에서 단어를 쉽게 분리할 수 있다.

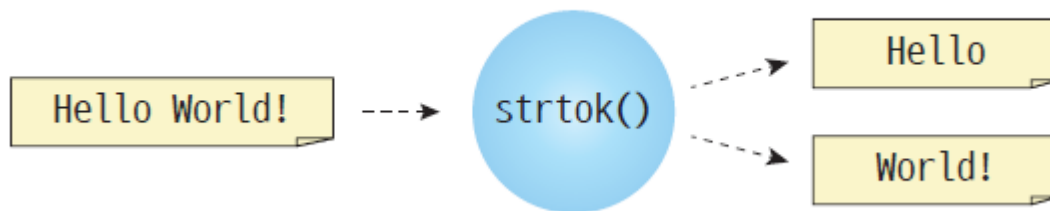
Syntax `strchr()`

예

```
char *p = strtok("Hello World!", " ");
```

분리자

문자열을 스페이스문자를 사용하여 단어들로 분리한다.



# 문자열 토큰 분리

- strtok() 함수를 사용하면 문장에서 단어를 쉽게 분리할 수 있다.
- 두번째부터 NULL을 입력하는 이유: strtok은 내부적으로 **이전 문자열의 상태를 기억하고**, 다음 토큰을 그 **저장된 위치로부터** 계속 찾기 때문입니다.

```
char[] s = "Man is immortal, because he has a soul";  
t1 = strtok(s, " ");           // 첫 번째 토큰: Man  
t2 = strtok(NULL, " ");        // 두 번째 토큰: is  
t3 = strtok(NULL, " ");        // 세 번째 토큰: immortal,  
t4 = strtok(NULL, " ");        // 네 번째 토큰: because
```

# 문자열 토큰 분리

```
// strtok 함수의 사용예
```

```
#include <string.h>
```

```
#include <stdio.h>
```

```
char s[] = "Man is immortal, because he has a soul";
```

분리자

```
char seps[] = " ,\t\n";
```

```
char *token;
```

```
int main( void )
```

```
{
```

```
    // 문자열을 전달하고 다음 토큰을 얻는다.
```

```
    token = strtok( s, seps );
```

```
    while( token != NULL )
```

```
    {
```

```
        // 문자열 s에 토큰이 있는 동안 반복한다.
```

```
        printf( "토큰: %s\n", token );
```

```
        // 다음 토큰을 얻는다.
```

```
        token = strtok( NULL, seps );
```

```
    }
```

```
}
```

토큰: Man  
토큰: is  
토큰: immortal  
토큰: because  
토큰: he  
토큰: has  
토큰: a  
토큰: soul

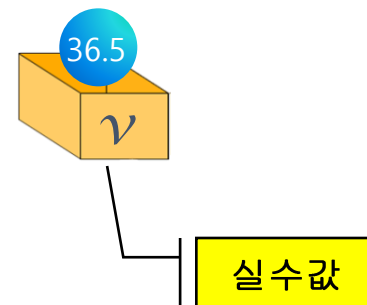
# 중간 점검

1. 문자열 s1를 문자열 s2로 복사하는 문장을 써라.
2. "String"을 저장하려면 최소한 어떤 크기 이상의 문자 배열이 필요한가?
3. 문자열을 서로 비교하는 함수는?
4. strcpy()와 strncpy()의 차이점은 무엇인가?
5. s1[]에 저장된 문자열 뒤에 s2[]를 붙이고 싶으면 어떤 라이브러리 함수를 어떻게 사용하여야 하는가?
6. strcmp("dog", "dog")의 반환값은 얼마인가?



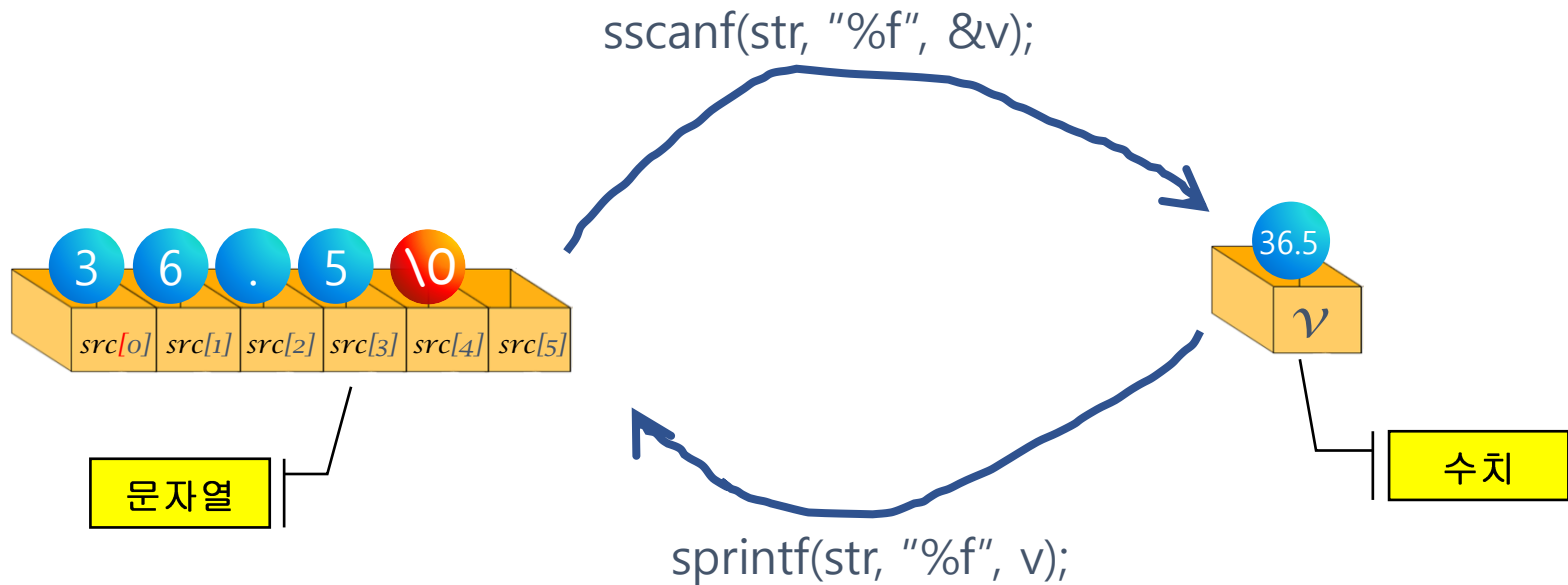
# 문자열 수치 변환

- 문자열 "36.5"와 수치값 36.5는 컴퓨터 안에서 상당히 다르게 저장된다.



# 문자열 변환: sprintf()와 sscanf()

- 앞에 붙은 s는 string 을 의미한다.





# 예제

```
#include <stdio.h>

int main( void )
{
    char instring[]="file 12";
    char name[10];
    int number;

    sscanf(instring, "%s %d", name, &number);

    printf("name = %s \n", name);
    printf("number = %d \n", number);
    return 0;
}
```

name = file  
number = 12

# 예제

```
#include <stdio.h>

int main( void )
{
    char buffer[50];
    int x=10, y=20, result;


    result = x + y;
    sprintf(buffer, "정수 %d와 정수 %d를 더하면 %d입니다.", x, y, result);

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

정수 10와 정수 20를 더하면 30입니다.

# Lab: 영상 파일 이름 자동 생성

- 파일 이름을 자동으로 생성하는 프로그램을 작성해보자.



```
image0.jpg  
image1.jpg  
image2.jpg  
image3.jpg  
image4.jpg  
image5.jpg
```

# 예제

```
#include <stdio.h>
#include <string.h>
```

```
int main(void)
```

```
{
```

```
    char filename[100];
```

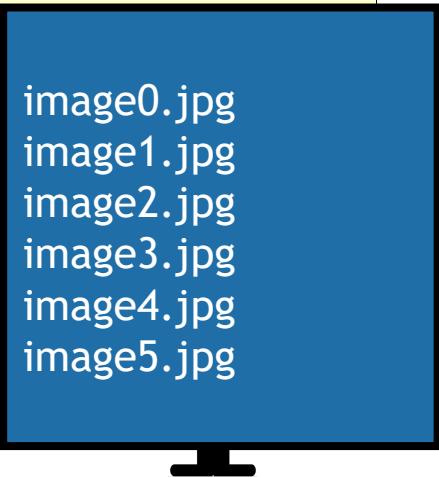
```
    int i;
```

```
    for(i=0; i < 6; i++){
        sprintf(filename, "image%d.jpg", i);
        printf("%s \n", filename);
    }
```

```
    return 0;
```

```
}
```

순차적인 파일  
이름을 만든다.

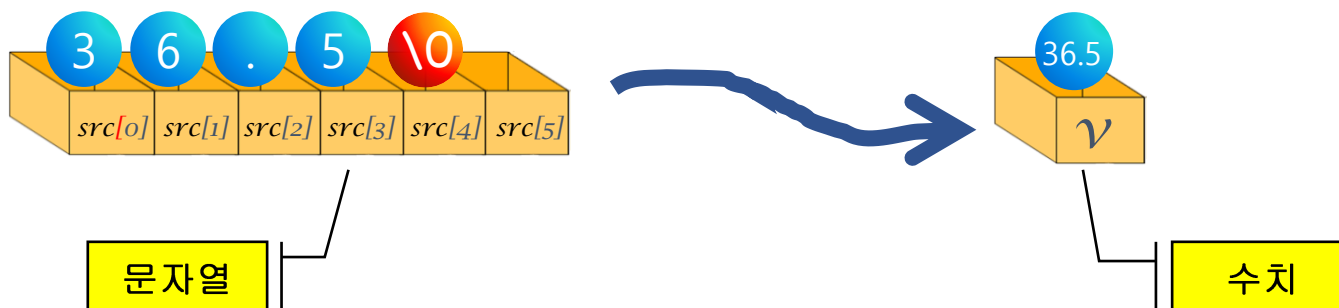


```
image0.jpg
image1.jpg
image2.jpg
image3.jpg
image4.jpg
image5.jpg
```

# 문자열을 수치로 변환하는 전용함수

- 전용 함수는 scanf()보다 크기가 작다.
- stdlib.h에 원형 정의- 반드시 포함

함수	설명
<code>int atoi( const char *str );</code>	str을 int형으로 변환한다.
<code>double atof( const char *str );</code>	str을 double형으로 변환한다.



# 문자열 수치 변환

```
#include <stdio.h>
#include <stdlib.h>

int main( void )
{
    char s1[] = "100";
    char s2[] = "12.93";
    int i;
    double d, result;

    i = atoi(s1);
    d = atof(s2);

    result = i + d;
    printf("연산 결과는 %.2f입니다.\n", result);

    return 0;
}
```

연산 결과는 112.93입니다.

# 중간 점검

1. 실수값 3.141592와 문자열 "3.141592"가 차지하는 메모리 공간을 비교하라.
2. 문자열 "3.141592"를 실수값을 변환하고자 할 때 사용할 수 있는 함수는 어떤 것들이 있는가?
3. printf()와 sprintf()가 다른 점은 무엇인가?



# 문자열의 배열

- 문자열이 여러 개 있는 경우에는 어떤 구조를 사용하여 저장하면 제일 좋을까?
  - (예) "init", "open", "close", ...
- 2가지의 방법이 있다.
  - 문자열 배열
  - 문자 포인터 배열

"init"

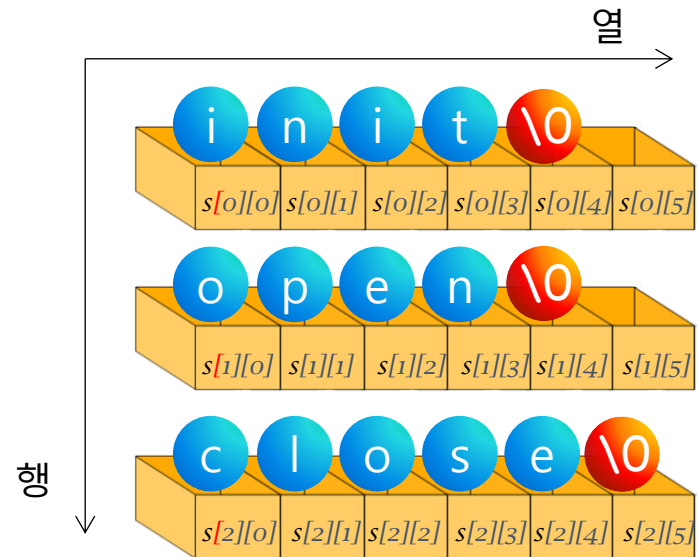
"open"

"close"



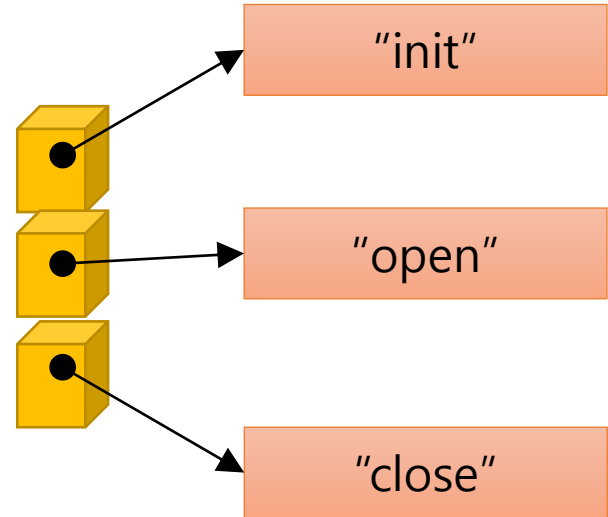
# 1. 문자열 배열

```
char s[3][6] = {  
    "init",  
    "open",  
    "close"  
};
```



## 2. 문자 포인터 배열

```
char *s[3] = {  
    "init",  
    "open",  
    "close"  
};
```



# 예제



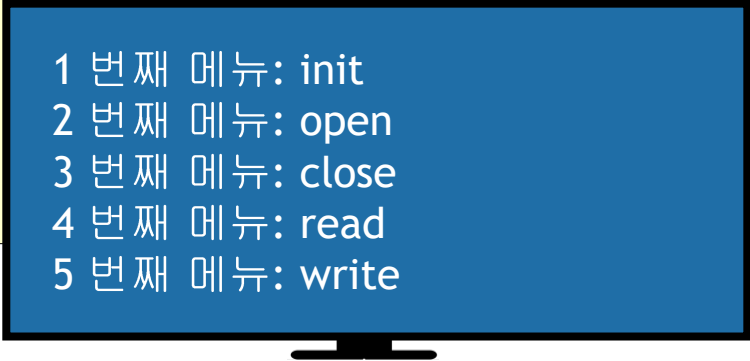
실습

```
#include <stdio.h>

int main(void)
{
    int i;
    char menu[5][10] = {
        "init",
        "open",
        "close",
        "read",
        "write"
    };

    for (i = 0; i < 5; i++)
        printf("%d 번째 메뉴: %s \n", i+1, menu[i]);

    return 0;
}
```



1 번째 메뉴: init  
2 번째 메뉴: open  
3 번째 메뉴: close  
4 번째 메뉴: read  
5 번째 메뉴: write

# 2차원 배열로 입력



실습

```
#include <stdio.h>
int main( void )
{
    int i;
    char fruits[3][20];
    for(i = 0; i < 3; i++) {
        printf("과일 이름을 입력하시오: ", fruits[i]);
        scanf("%s", fruits[i]);
    }
    for(i = 0; i < 3; i++)
        printf("%d번째 과일: %s\n", i, fruits[i]);
    return 0;
}
```

```
과일 이름을 입력하시오: 사과
과일 이름을 입력하시오: 배
과일 이름을 입력하시오: 포도
0번째 과일: 사과
1번째 과일: 배
2번째 과일: 포도
```

# Lab: 한영 사전의 구현

- 3차원 문자열 배열을 이용하여 간단한 한영 사전을 구현하여 보자.

단어를 입력하시오:boy  
boy: 소년



# 한영 사전 구현

```
#define ENTRIES 5

int main( void )
{
    int i, index;
    char dic[ENTRIES][2][30] = {
        {"book", "책"},
        {"boy", "소년"},
        {"computer", "컴퓨터"},
        {"lanuguage", "언어"},
        {"rain", "비"},
    };
    char word[30];
```

# 한영 사전 구현

```
printf("단어를 입력하시오:");  
scanf("%s", word);  
  
index = 0;  
for(i = 0; i < ENTRIES; i++)  
{  
    if( strcmp(dic[index][0], word) == 0 )  
    {  
        printf("%s: %s\n", word, dic[index][1]);  
        return 0;  
    }  
    index++;  
}  
printf("사전에서 발견되지 않았습니다.\n");  
}
```

단어를 입력하시오:book  
book: 책

# Q & A

