
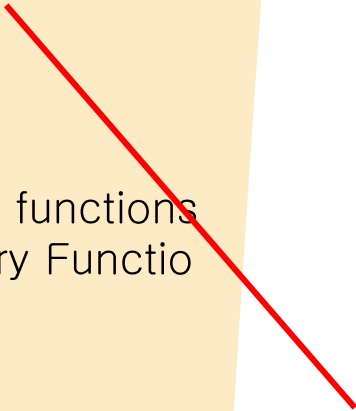


# Ch.12 Characters and Strings

# What you will learn in this chapter

- 
- How to express characters
  - How to represent strings
  - What is a string ?
  - Input/output of strings
  - Character processing library functions
  - Standard Input/Output Library Functions
- 

Since humans use characters to express information, strings occupy an important position in programs . In this chapter, we will take a closer look at string handling methods in C.



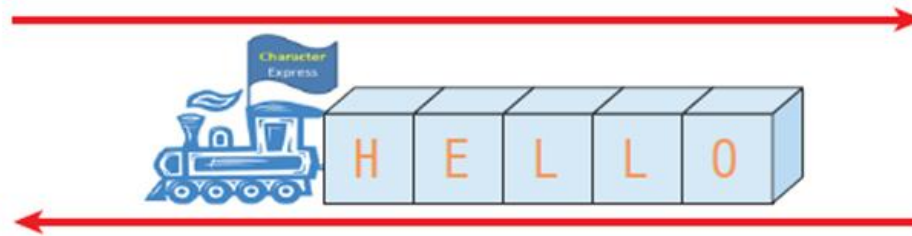
# The importance of letters

- Text is of utmost importance to humans .

A lot of computer work is done using text.

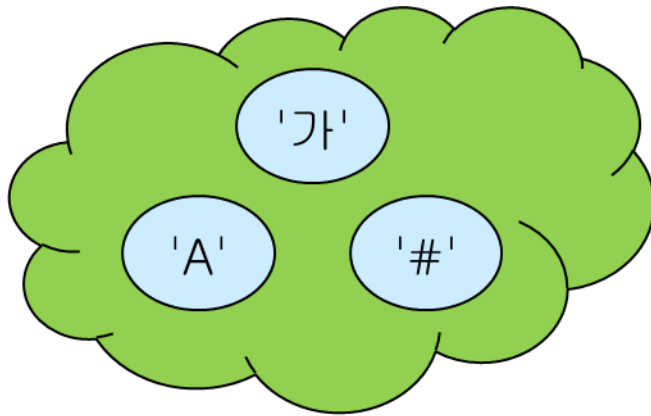


computer

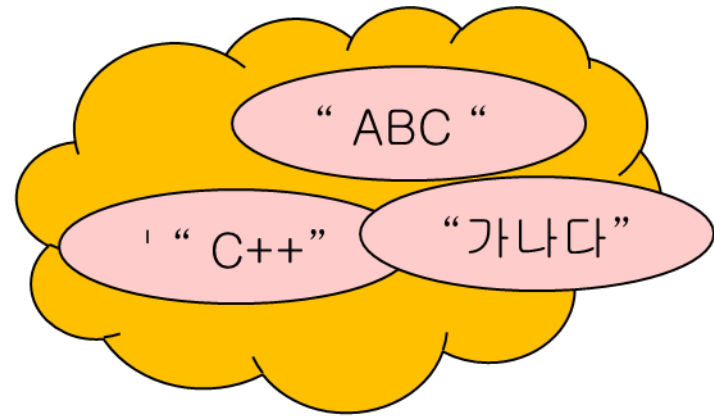


user

# Characters and strings



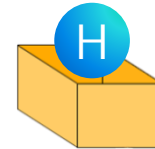
Characters



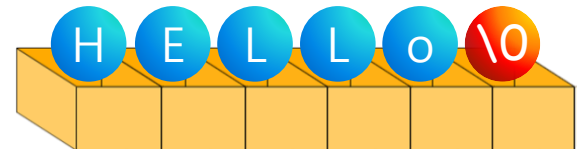
Strings

# How to represent strings

- *string* : A collection of several characters
  - "A "
  - "Hello World!"
  - " The value of variable score is %d "
- *String variable*
  - A variable that can store a mutable string.
  - Where is a good place to save it ?



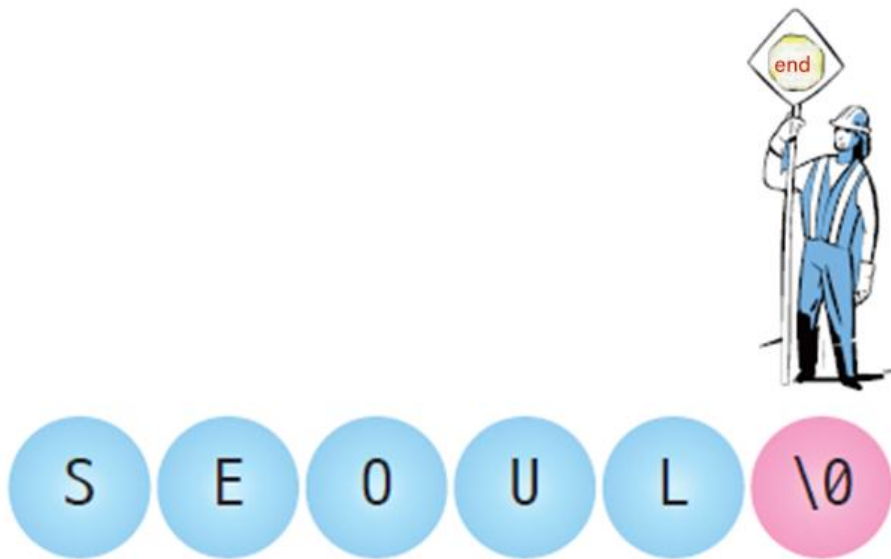
One character is stored in a char type variable.



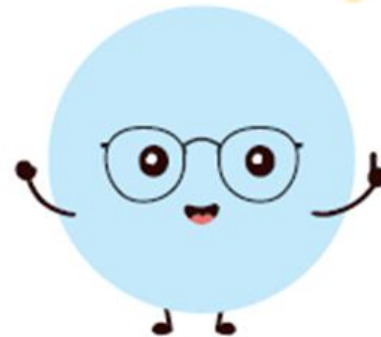
*Strings are stored as char type arrays*

# NULL character

- NULL character : Indicates the end of a string .

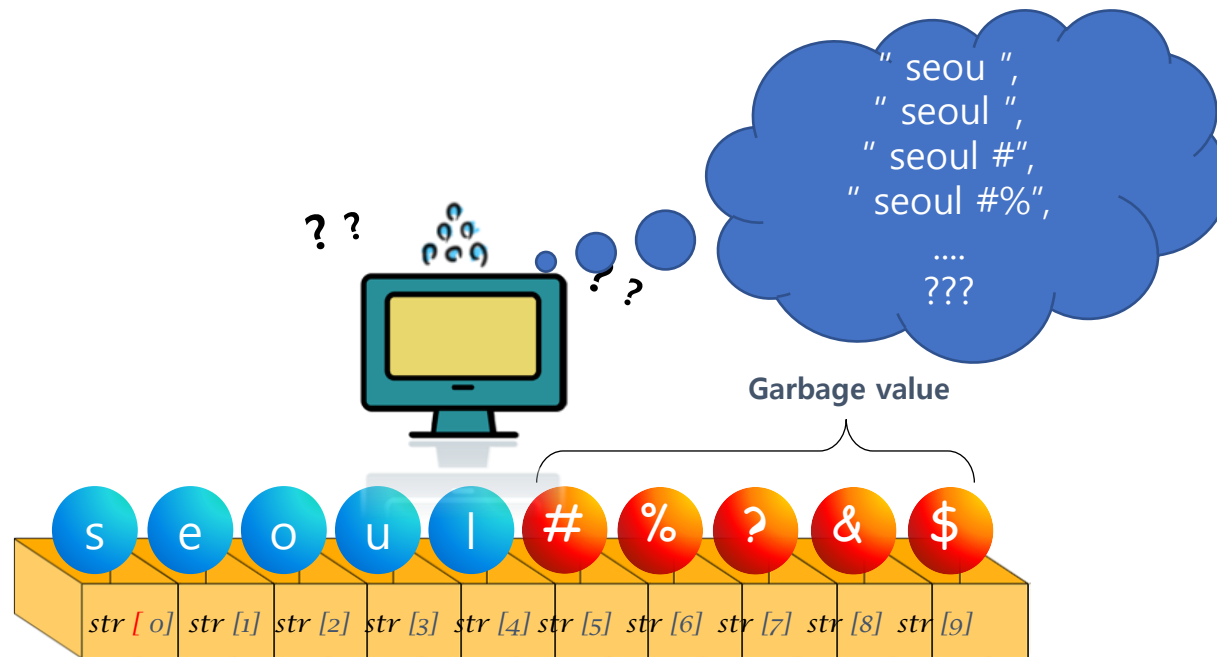


The NULL character indicates the end of a string.



# Why do we need to mark the end of a string ?

- Since we don't know where the string ends, we need to mark it .



# Example #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;
```

```
}
```

Let's print out the characters in the character array one by one . When printing a string stored in a character array, you can use %s , but here, in order to experience the basic method of string processing, we print out the characters in the character array one by one on the screen and stop the repetition when a NULL character is encountered .



abc



# reference

## Note

Let's take a quick C language quiz here. C language code

Let's think about the difference between A, 'A', and "A".

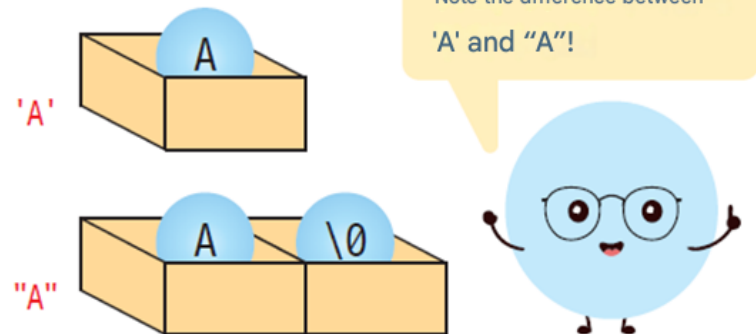
**A:** The compiler considers A as the name of a variable.

**'A':** Represents the letter A.

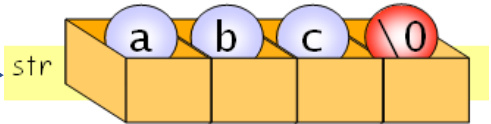
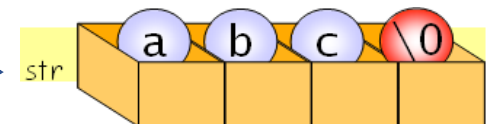
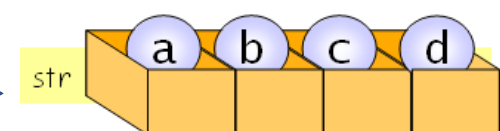
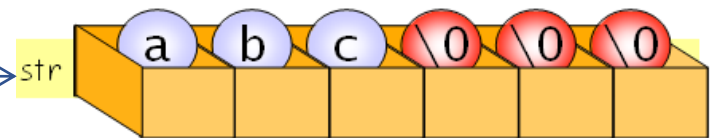
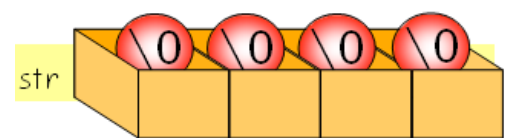
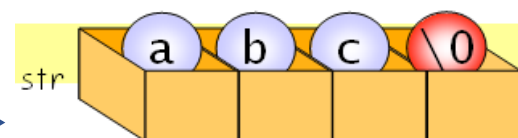
**"A":** Represents a string consisting only of the letter A. This is different from 'A'.

The thing to note here is the difference between 'A' and "A". 'A' represents a single character and is the same as the ASCII code for the character A.

A. "A" is a string, and the NULL character is added to the ASCII code of A to indicate the end of the string.

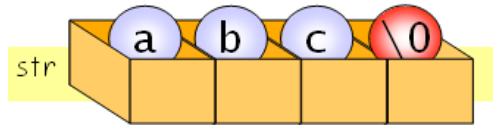


# Initializing a character array

- `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[];` Error → The compiler doesn't know how much memory to allocate.

# Output of string



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

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

A blue computer monitor with a black frame and a black base. The screen displays the text 'abc' in white. Two red, jagged lines connect the code blocks on the left to the monitor, indicating the output of the programs.

abc

# Example #2

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

```
int main( void )
```

```
{
```

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

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

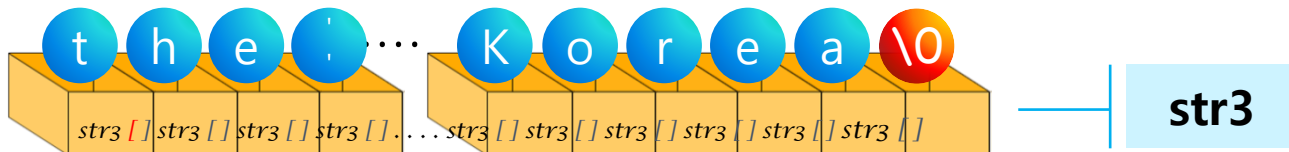
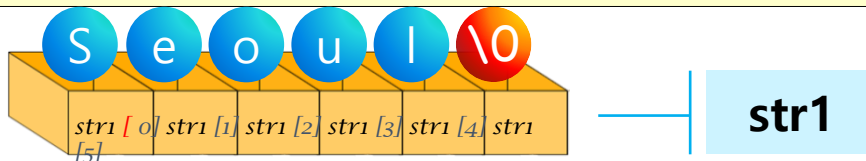
```
    char str3[] = "the capital city of Korea." ;
```

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

```
    return 0;
```

```
}
```

Seoul is the capital city of Korea.



# Example #3

```
#include <stdio.h>

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

    int i ;
    printf ( " Original string =%s\n" , src );
    for (i = 0; src[i] != '\0' ; i++)
        dst [ i ] = src [ i ];
    dst [ i ] = '\0' ;
    printf ( " Copied string =%s\n" , dst );

    return 0;
}
```

NULL and '\0' are the same .

Original string =Action speaks louder than words  
Copied string =Action speaks louder than words

# Example of calculating string length

```
// Program to find the length of a string
#include <stdio.h>

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

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

    printf ( " The length of the string \"%s\" is %d .\n" , str , i );
    return 0;
}
```

String "C language is The length of "easy" is 18.

# How to change the character array

1. The first method is to individually assign the desired character to each array element. This is a surefire way, but very inconvenient .

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

2. You can copy a string to a character array using the library function `strcpy ()` .

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

# Wrong way

3. The following method , which seems to be the most convenient , cannot be used . Please be careful .

```
char str[10] = "Hello";  
str = "World"; // Grammatical error !
```

As we learned in arrays, the name of an array is a pointer constant that points to the array. It cannot be changed.





# String constant

- String constant : Strings included in the program source code such as "HelloWorld".
- String Constants are stored in **text segment** among the memory areas.

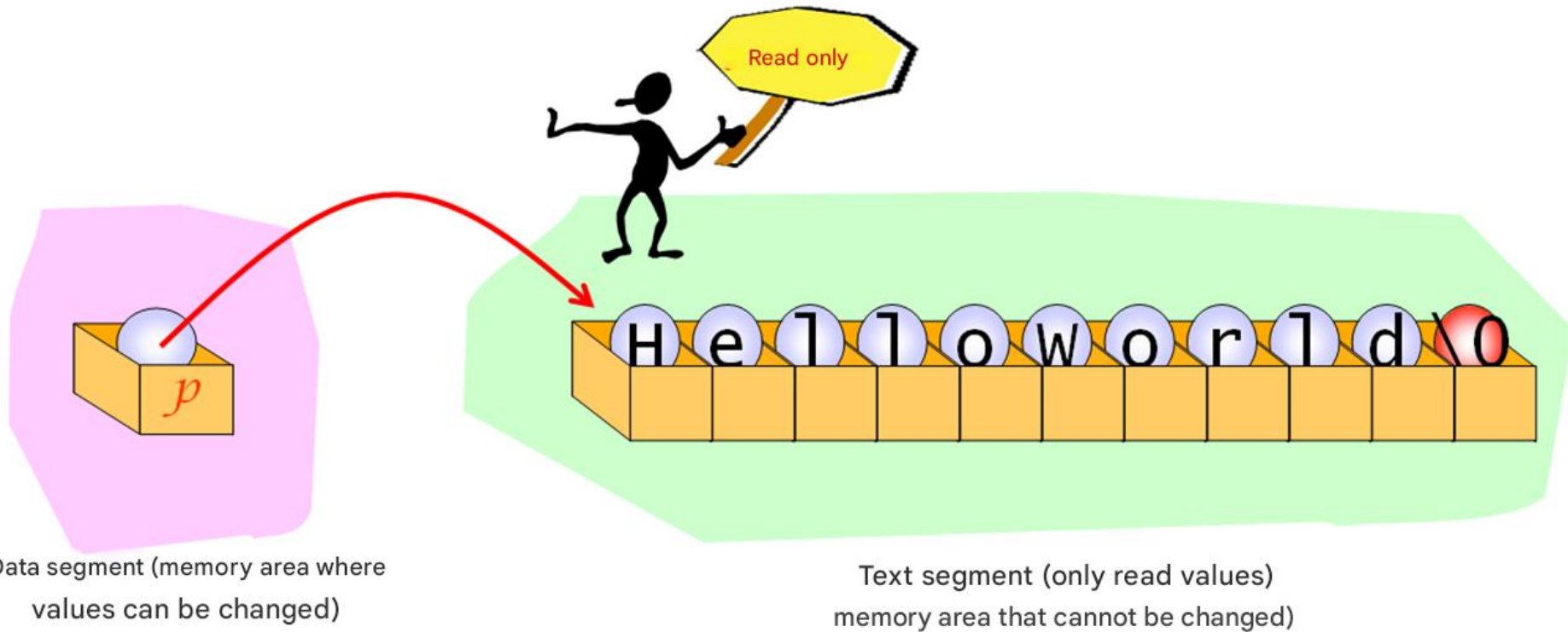
```
char *p = " HelloWorld ";
```

What exactly does the above sentence mean ?



# String constant

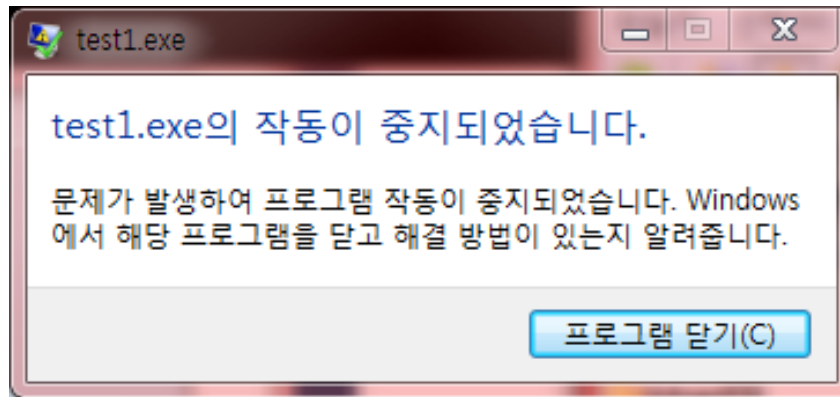
```
char *p = " HelloWorld ";
```



# String constant

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

An error occurs when trying to store c  
haracters in a text segment via p .

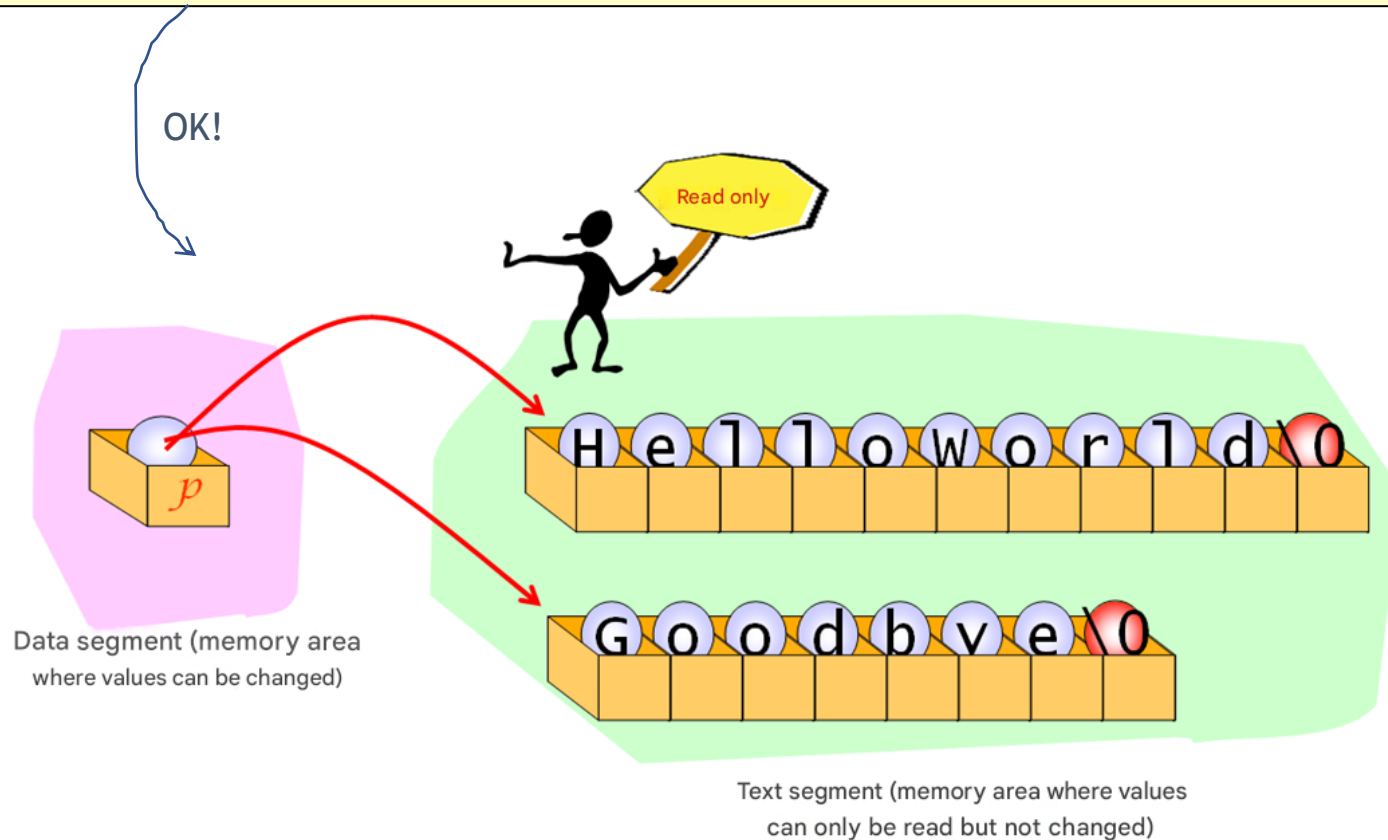


Text segments cannot  
be changed .



# String constant

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



# Example

```
#include < stdio.h >

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

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

    p = "Goodbye" ; // possible
    printf ( "%s \n" , p);
    // p[0] = 'a'; // An error occurs .

    return 0;
}
```

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

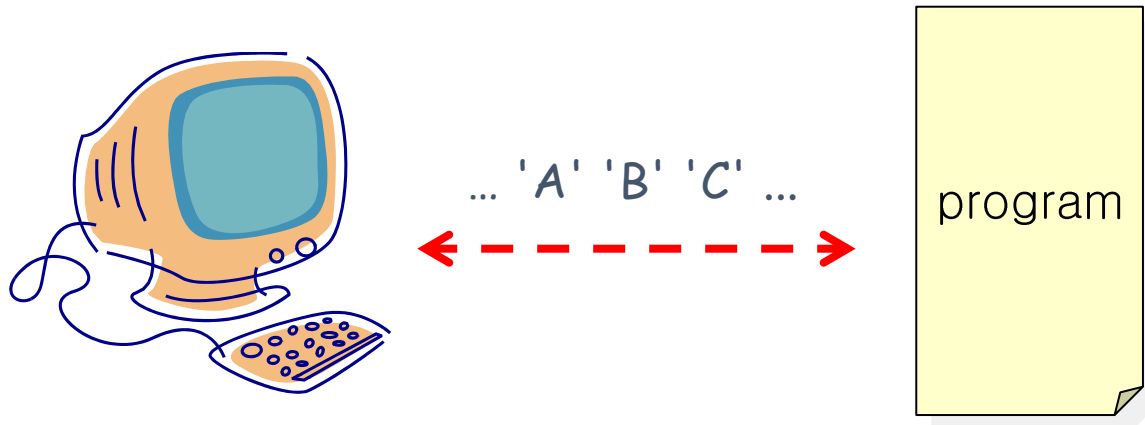
# Check points

1. How are strings defined in C ?
2. What is the role of the NULL character in a string ?
3. What is the ASCII code value of the NULL character ?
4. What happens if I print a string that doesn't end with a NULL character ?
5. Explain the difference between B, 'B', and "B" .
6. Where are mutable strings stored ?
7. Why is the size of the character array one larger than the size of the string ?
8. How to initialize a character array to a string.

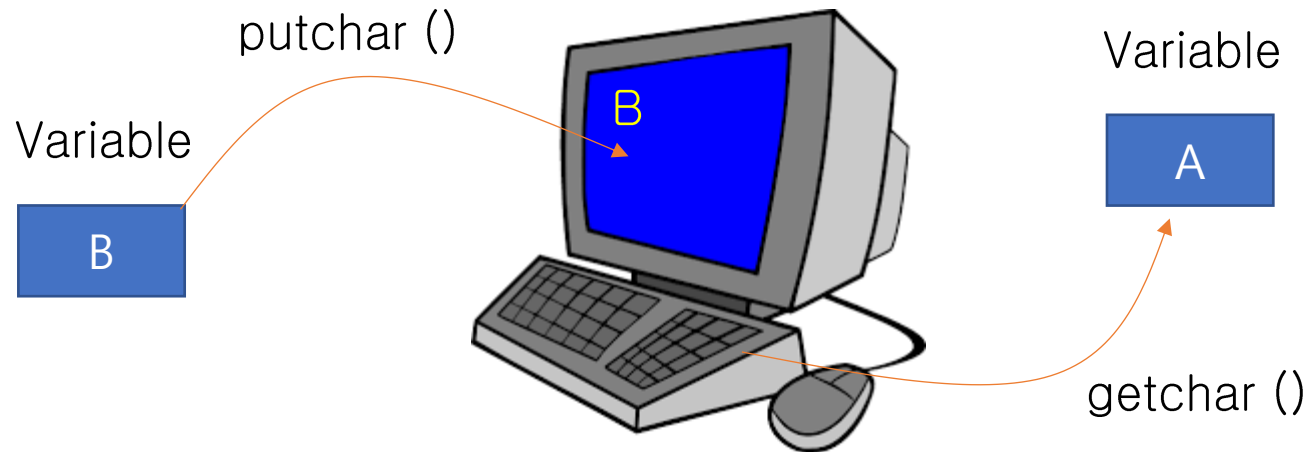


# Character input/output library

Input/output functions	explanation
<code>int getchar(void)</code>	Reads and returns a single character.
<code>void putchar(int c)</code>	Prints the character stored in variable <code>c</code> .
<code>int _getch(void)</code>	Reads and returns a single character ( <b>without using a buffer</b> ).
<code>void _putch(int c )</code>	Prints the character stored in variable <code>c</code> ( <b>without using a buffer</b> ).
<code>scanf("%c", &amp;c)</code>	Read one character and store it in variable <code>c</code> .
<code>printf("%c", c);</code>	Prints the character stored in variable <code>c</code> .



# getchar (), putchar ()





# getchar(), putchar()

```
// Usage of
#include <stdio.h>
int main( void )
{
    int ch ; // Note the use of integers
    while ( ( ch = getchar () ) != EOF )
        putchar ( ch );
    return 0;
}
```

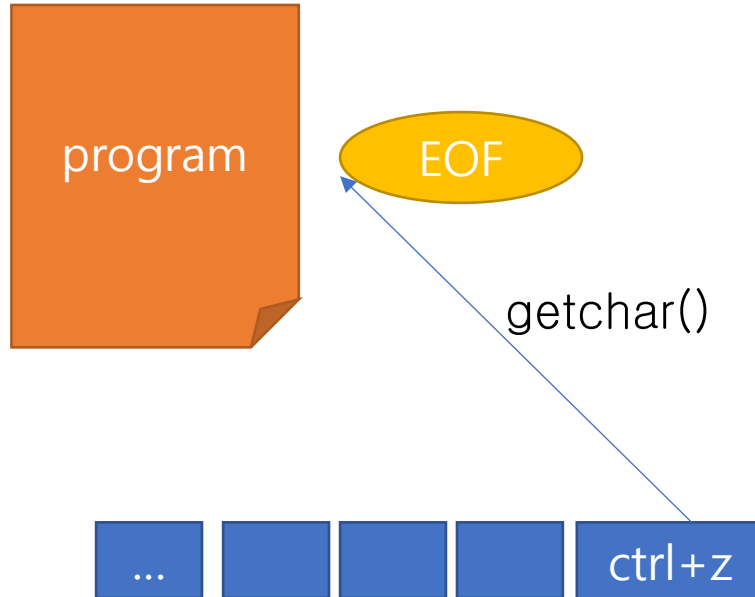
Character indicating  
End Of File ,  
EOF is an integer -1 .



a  
a  
b  
b  
^Z

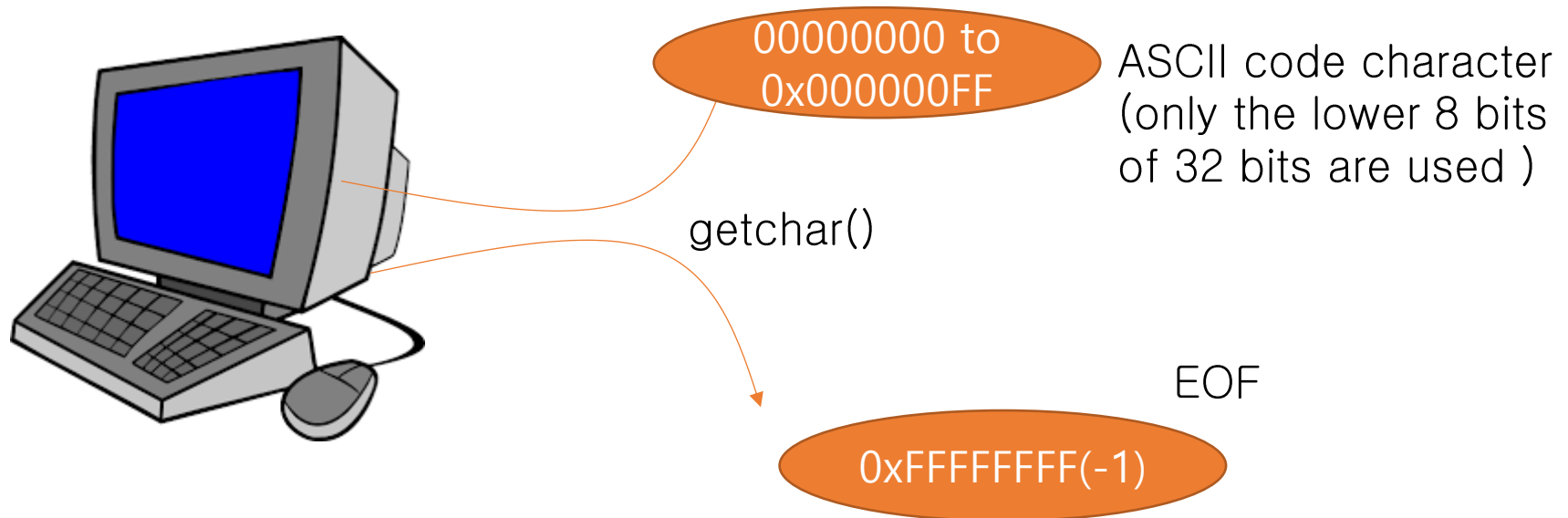
# EOF

- EOF(End Of File): A special symbol indicating the end of input.



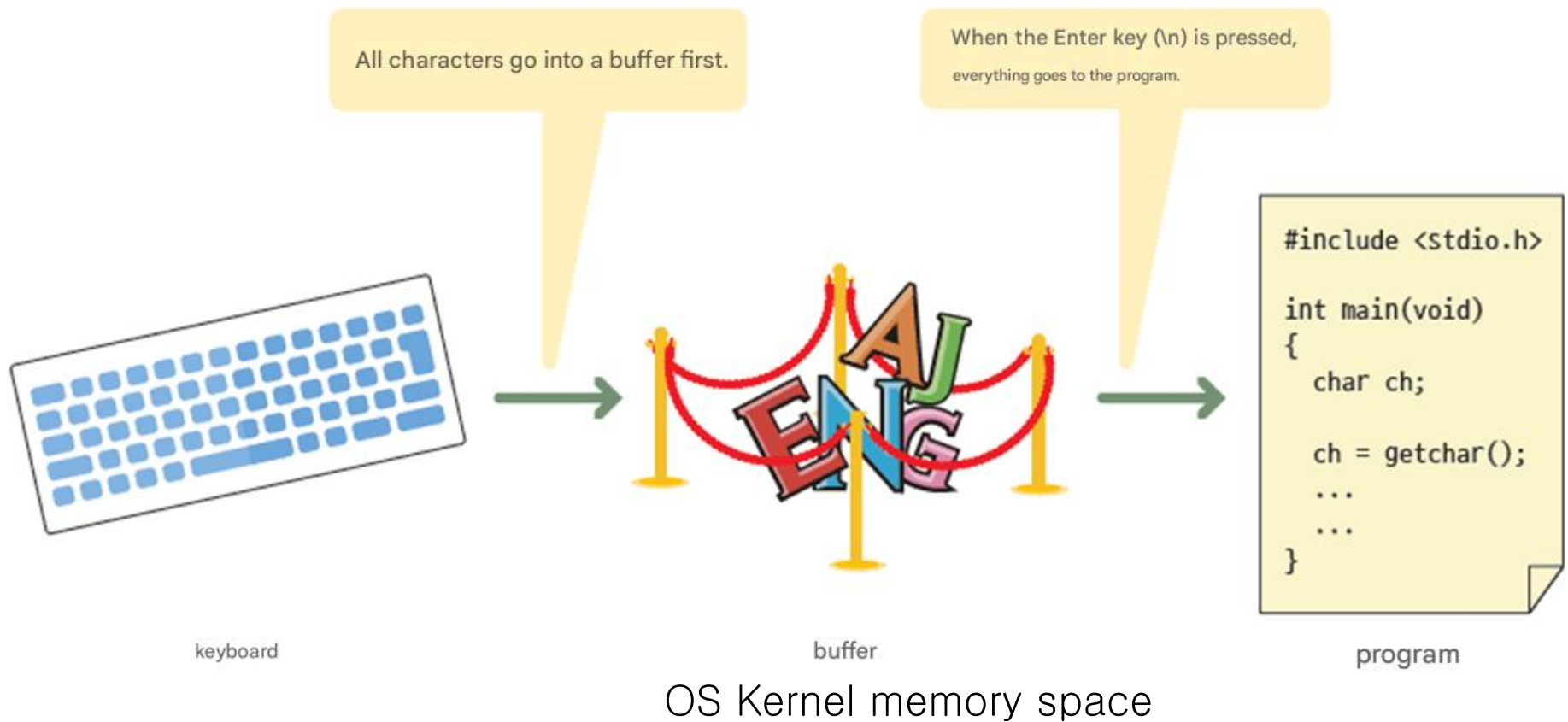
# The return value of getchar() Why it's an integer

- The return type of getchar() is "int" to distinguish the ASCII code from EOF.
- If you use a 32-bit int type, you can clearly distinguish between the ASCII code (0x00000000 to 0x000000FF) and EOF (0xFFFFFFFF), which is a 32-bit int type -1. (It's impossible to do with 8 bits).



# Buffering

- press Enter to get input?



# `_getch()`, `_getche()`, `getchar()`

	header file	Whether to use buffer	Echo or not	Responsiveness	Whether to modify text
<code>getchar()</code>	<code>&lt;stdio.h&gt;</code>	used (Entered by pressing Enter)	echo	line unit	possible
<code>_getch()</code>	<code>&lt;conio.h&gt;</code>	Not used	Does not echo	character unit	impossibility
<code>_getche()</code>	<code>&lt;conio.h&gt;</code>	Not used	echo	character unit	impossibility



Choose the one  
that suits your  
purpose !

If you want to  
receive it  
immediately  
without buffering,  
use `_getch()` .

# \_ getch(), \_ putch()

Useful for editing programs!

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

```
int main( void )
```

```
{
```

```
    int ch ;
```


```
    while (( ch = _ getch () ) != 'q' )
```

```
        _ putch( ch );
```

```
    return 0;
```

```
}
```

Do not use buffers ,  
No echo !



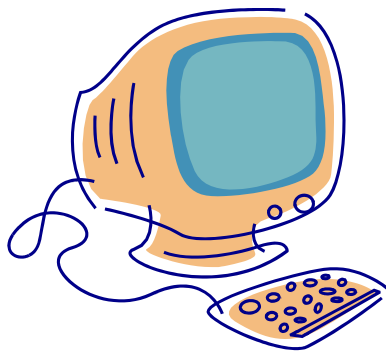
abc



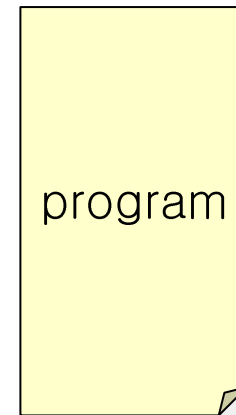
# String input/output library functions

Windows

Input/output functions	explanation
<code>int scanf("%s", s)</code>	Read a string and store it in a character array
<code>int printf("%s", s)</code>	Prints the string stored in array <code>s[]</code> .
<code>char * gets_s(char *s, int size)</code>	Read a line of string and store it in the character array <code>s[]</code> .
<code>int puts(const char *s)</code>	Prints a single line of string stored in array <code>s[]</code> .



... Hello World!...



# Example

```
#include <stdio.h>

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

    printf ( "Enter your name : " );
    scanf ( "%s" , name);
    printf ( "Enter your current address : " );
    scanf ( "%s" , address);

    printf ( "Hello , Mr. % s who lives in %s .\n" , address, name);
    return 0;
}
```

The address is not fully saved.

```
Enter your name : Hong Gil-dong
Enter your current address : 1 Jongno -gu, Seoul

Hello . I am Hong Gil-dong living in Seoul .
```



# gets\_s () and puts() string input/output

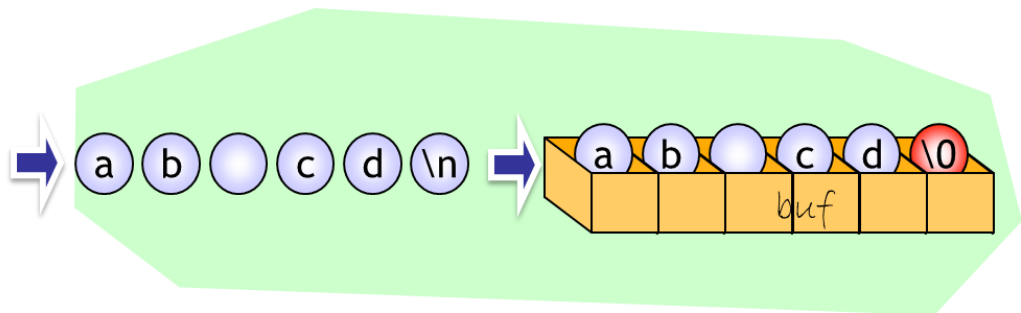
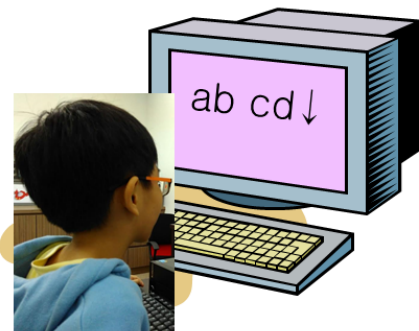
## Syntax gets\_s()

yes

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

Takes one line of input from the user.

Prints one line.



gets\_s(buf, 100);

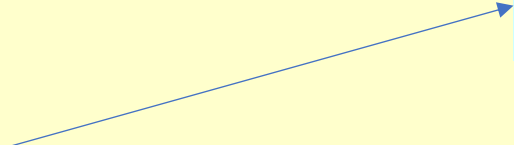
# Example

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

    printf ( "Enter your name : " );
    gets_s ( name, sizeof (name));
    printf ( "Enter your current address : " );
    gets_s ( address, sizeof (address));

    printf ( "Hello , Mr. %s who lives in %s .\n" , address, name);
    return 0;
}
```

Used when entering  
more than one word .



```
Enter your name : Hong Gil-dong
Enter your current address : 1 Jongno -gu, Seoul

Hello ? I am Hong Gil-dong, living at 1 Jongno-gu, Seoul .
```

# Character processing library functions

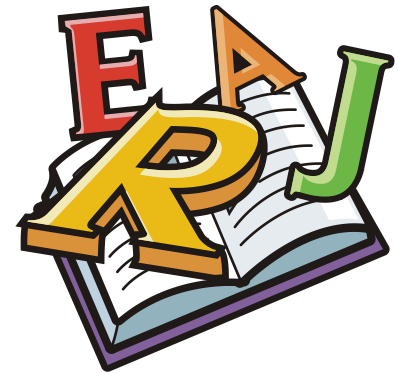
- Check or convert characters .

Function	explanation
isalpha (c)	Is c an English letter ? (az, AZ)
isupper (c)	Is c uppercase ? (AZ)
islower(c)	c lowercase ? (az)
isdigit (c)	Is c a number ? (0-9)
isalnum(c)	c a letter or a number ? (az, AZ, 0-9)
isxdigit(c)	Is c a hexadecimal number ? (0-9, AF, af)
isspace(c)	Is c a space character ? (' ', '\n', '\t', '\v', '\r')
ispunct(c)	Is c a punctuation character ?
isprint(c)	C a printable character ?
iscntrl(c)	Is c a control character ?
isascii(c)	c an ascii code ?

# Character processing library functions

- Check or convert characters .

Function	explanation
<code>toupper(c)</code>	c to uppercase .
<code>tolower(c)</code>	c to lowercase .
<code>toascii(c)</code>	c to ASCII code .



# Example

```
#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;
```

```
}
```

Check if it is lowercase

Convert to uppercase

```
abcdef
ABCDEF
^ Z
```

To input EOF from your keyboard, type ^Z

# Lab: Words century

- Let's write a program that counts the number of words in a string. If the string is "the c book..." , the output will be as follows :

A computer monitor with a black frame and a black stand. The screen is blue and displays the text "Number of words : 3" in white.

Number of words : 3

# Example

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

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

    return 0;
}
```

# Example

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

    for ( i = 0; s [ i ] != NULL ; ++ i ) // Examine each letter of
        if ( isalpha ( s [ i ] )) // If the letter of
        {
            if (waiting) // If you are waiting for a word
            {
                wc ++; // increment the counter
                waiting = 0; // Processing words
            }
        }
        else // if it is not an alphabet
            waiting = 1; // Wait for a word .

    return wc ;
}
```



# Lab: Check valid password

- A valid password must be at least 7 characters long , contain at least one lowercase letter, at least one uppercase letter, and at least one number .

Enter your password : abc1234

The password is not valid .

# Solution

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

int main( void )
{
    int lower_case_count = 0; // Number of lowercase letters
    int upper_case_count = 0; // Number of uppercase letters
    int digit_count = 0; // number of digits
    char pass[100];
    int len ;

    printf ( " Enter your password : " );
    gets_s (pass, sizeof (pass));
```

# Solution

```
len = strlen (pass); // length of the string
if ( len < 7) {
    printf ( " Not a valid password .\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 ( " This is a strong password . \n" );
else
    printf ( " Not a valid password \n" );
return 0;
}
```

# Check points

1. What header files must I include to use the character processing library functions ?
2. What is the difference between `getchar()` and `getch()` ?
3. What is the return value of `ispunct('.')` ?
4. What is the return value of `toupper('a')` ?



# String processing library

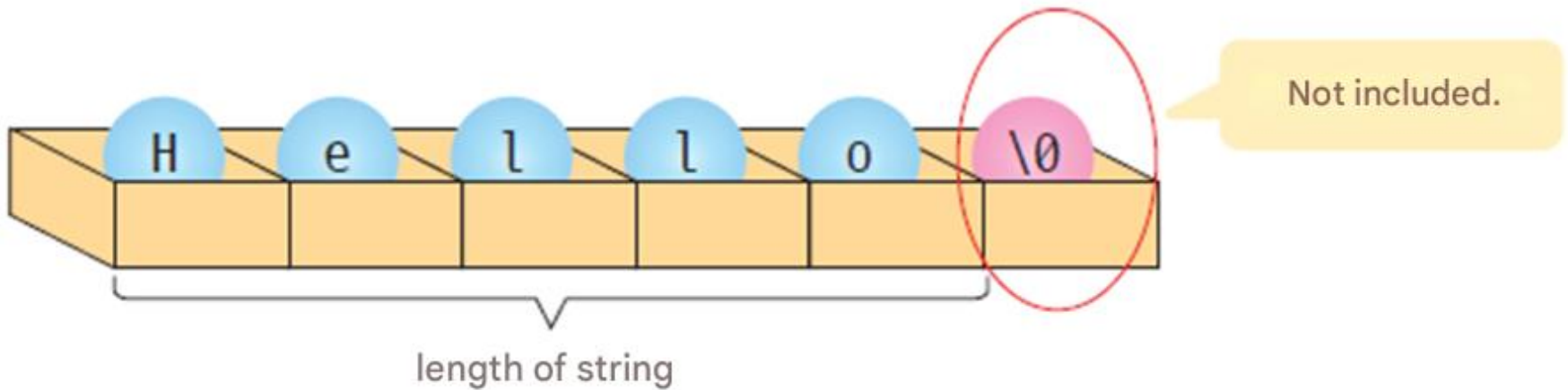
- String functions are declared in `string.h`. Therefore, to use these functions, you must include `string.h` at the beginning of your program.

Function	explanation
<code>strlen(s)</code>	Finds the length of string <code>s</code> .
<code>strcpy(s1, s2)</code>	Copy <code>s2</code> to <code>s1</code> .
<code>strcat(s1, s2)</code>	Paste <code>s2</code> to the end of <code>s1</code> .
<code>strcmp(s1, s2)</code>	Compare <code>s1</code> and <code>s2</code> .
<code>strncpy(s1, s2, n)</code>	Copy at most <code>n</code> characters from <code>s2</code> to <code>s1</code> .
<code>strncat(s1, s2, n)</code>	Paste at most <code>n</code> characters from <code>s2</code> to the end of <code>s1</code> .
<code>strncmp(s1, s2, n)</code>	Compares <code>s1</code> and <code>s2</code> up to <code>n</code> characters .
<code>strchr(s, c)</code>	Finds the character <code>c</code> in string <code>s</code> .
<code>strstr(s1, s2)</code>	Find string <code>s2</code> in string <code>s1</code> .

# Length of string

- The function to calculate the length of a string is `strlen` (`const char *str`) .

`int size = strlen ("Hello"); // size becomes 5 .`



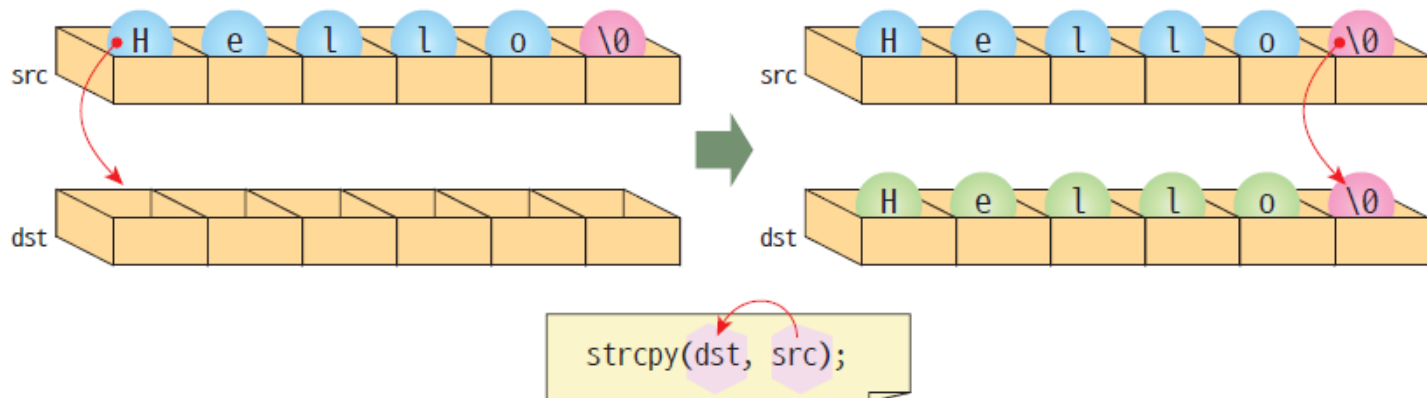
# Copy string

**Syntax** strcpy()

yes

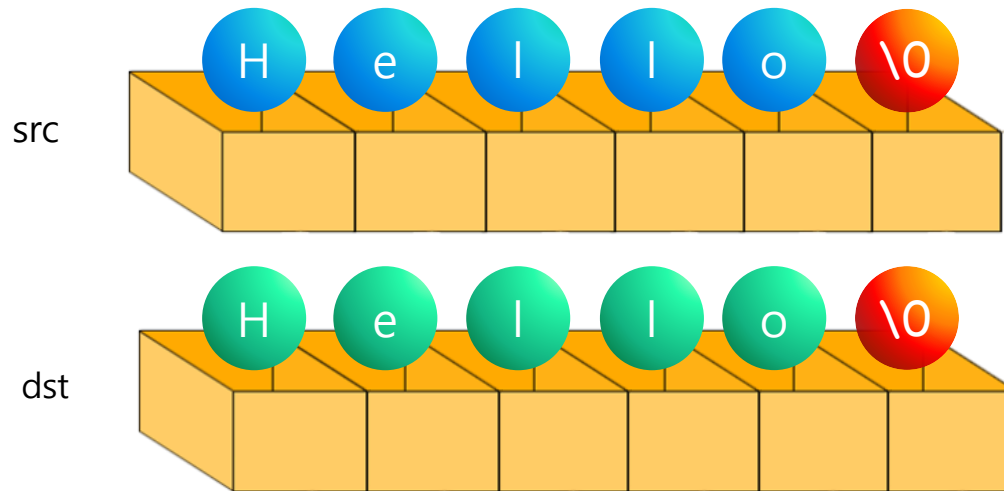
```
char dst[6];  
char src[6]="Hello";  
strcpy(dst, src);
```

// Copy src to dst.



# String copy animation

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





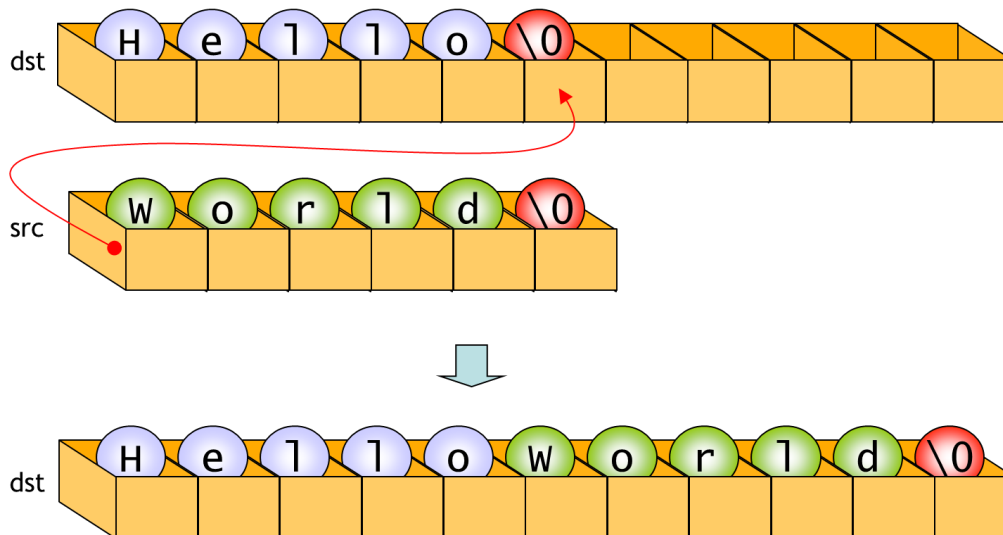
# String concatenation

Syntax `strcat()`

yes

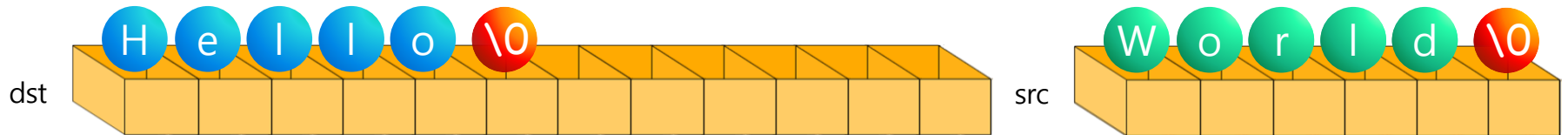
```
char dst[12]= "Hello";  
char src[6] = "World";  
strcat(dst, src);
```

// dst becomes "HelloWorld".



# String concatenation animation

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



# Example

```
// strcpy and 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 !

# String comparison

## Syntax

`strcmp()`

`strcmp()` compares strings `s1` and `s2` lexicographically.

If `s1` comes first in the (lexicographic) order, the negative number is

returned, 0 is returned if it is equal, and a positive number is returned if it is later.

yes

```
int result = strcmp("dog", "dog"); // 0 is returned.
```

return value	Relationship between <code>s1</code> and <code>s2</code>
$< 0$	<code>s1</code> comes before <code>s2</code> .
$0$	<code>s1 == s2</code>
$> 0$	<code>s1</code> comes after <code>s2</code> .

The string If they are equal,  
`strcmp ( )` returns



# Example

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


int main( void )
{
    char s1[80]; // first The word To save Character array
    char s2[80]; // second The word To save Character array
    int result;

    printf ( " first The word Enter :" );
    scanf ( "%s" , s1);
    printf ( " second The word Enter :" );
    scanf ( "%s" , s2);
```

# Example

```
result = strcmp (s1, s2);  
if ( result < 0 )  
    printf ( "% s is greater than % s" in front There are .\n" , s1, s2);  
else if ( result == 0 )  
    printf ( "% s is % s and are the same .\n" , s1, s2);  
else  
    printf ( "% s is greater than % s" behind are .\n" , s1, s2);  
return 0;  
}
```

Alphabetically, meaning it is  
at the front in the dictionary



Enter the first word : cat  
Enter the second word : dog  
cat comes before dog.

# Text search

- To check whether a given string contains a specific character, use `strchr()` .

Syntax

`strchr()`

예

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

'g' 문자의 주소를 반환한다.

# Text search

```
#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 ( "%c not found \n" , c );
```

```
    else {
```

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

```
        printf ( " First %c in %s found in %d \n" , c, s, loc);
```

```
    }
```

```
    return 0;
```

```
}
```

Search the letter c in string s



First g in language was found in 3



# String search

- To search for a specific string within a given string, use `strstr()`.

## Syntax

`strstr()`

yes

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

The `strstr()` function searches for a substring `sub` in a string `s`. If the substring is found, the address of that location is returned. If the substring is not found, a `NULL` value is returned.


# String search

```
#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 ( "%s not found \n" , sub );
    else {
        loc = ( int )(p - s);
        printf ( " First %s in % s found in %d\n" , sub , s, loc );
    }
    return 0;
}
```

Search sub string in string s



First bird in A bird in hand is worth two in the bush was found in 2

# String tokenization

- The strtok () function makes it easy to separate words from a sentence.

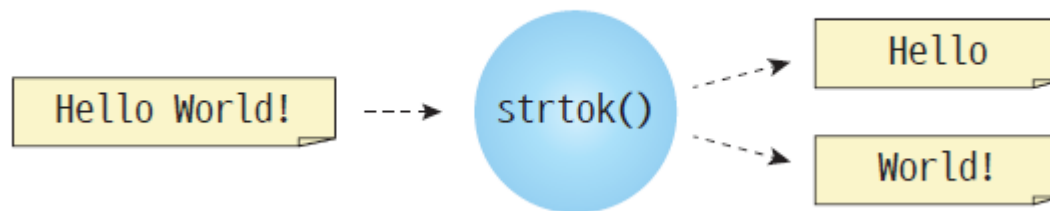
**Syntax** `strchr()`

yes

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

separator

Splits a string into words using spaces.



# String tokenization

- The strtok () function makes it easy to separate words from a sentence.

```
char[] s = "Man is immortal, because he has a soul" ;  
t1 = strtok (s, " "); // first th token : Man  
t2 = strtok (NULL, " "); // Second token : is  
T3 = strtok (NULL, " "); // Third token : immortal,  
t4 = strtok (NULL, " "); // fourth token : because
```

# String tokenization

```
// strtok Example of using a function
```

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


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

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

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

```
char *token;
```

separator



```
int main( void )
```

```
{
```

```
    // string And pass it on next Token Get .
```

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

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

```
    {
```

```
        // string In s The token present during Repeat .
```

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

```
        // next Token Get .
```

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

```
    }
```

```
}
```

```
Token : Man
Token : is
Token : immortal
Token : because
Token : he
Token : has
Token : a
Token : soul
```

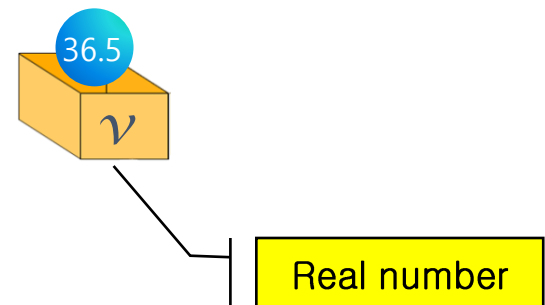
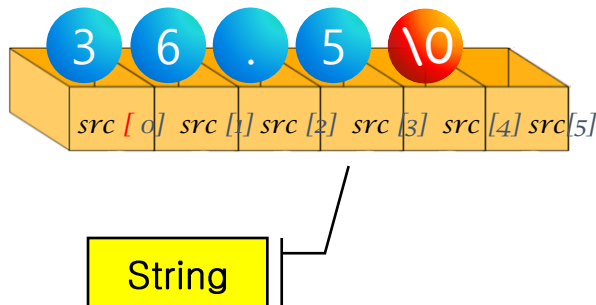
# Check points

1. Write a statement that copies string s1 to string s2.
2. What is the minimum size of a character array required to store a "String" ?
3. What is the function to compare strings ?
4. What is the difference between strcpy() and strncpy() ?
5. s2[] to the end of the string stored in s1[], what library function should I use and how ?
6. What is the return value of strcmp ("dog", "dog") ?



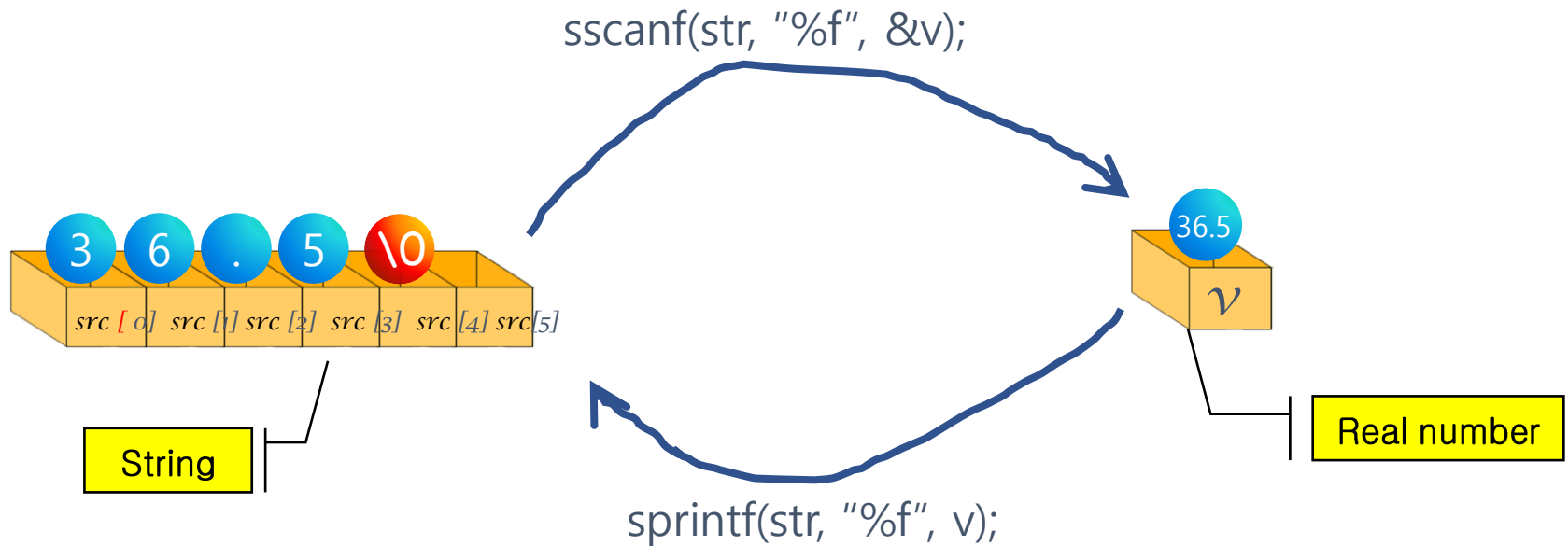
# String to number conversion

- String "36.5" and numeric value 36.5
- They are stored quite differently within a computer.



# sprintf () and sscanf ()

- in front The attached s means string .





# Example

```
#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

# Example

```
#include <stdio.h>

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

    result = x + y;
    sprintf (buffer, " Integer %d plus integer %d is %d .", x, y, result);

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

Integers 10 plus integer 20 is 30.

# Lab: Automatically generate video file names

- Let's write a program that automatically generates file names .

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

# Example

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

```
int main( void )
```

```
{
```

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

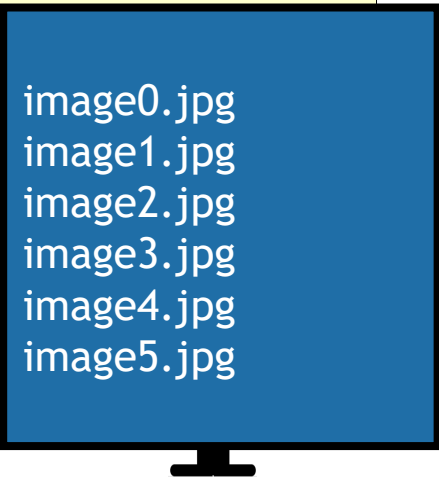
```
    int i ;
```

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

```
    return 0;
```

```
}
```

*Create sequential  
file names .*

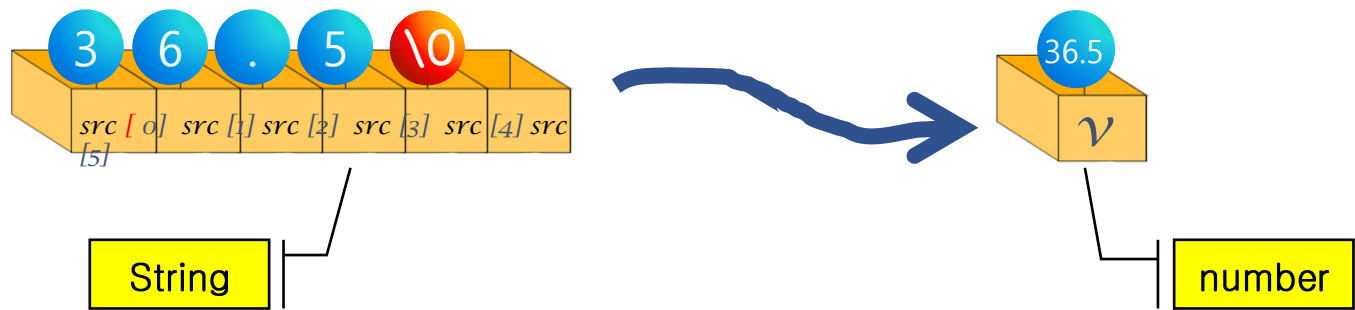


```
image0.jpg
image1.jpg
image2.jpg
image3.jpg
image4.jpg
image5.jpg
```

# Functions to convert a string to a number

- The dedicated function is smaller than scanf() .
- Prototype definition in stdlib.h - must be included

Function	explanation
<code>int atoi( const char *str );</code>	Convert string to int type .
<code>double atof( const char * str );</code>	Convert string to double type .



# String to number conversion

```
#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 ( " The result of the operation is %.2f .\n" , result);

    return 0;
}
```

The result of the operation is  
112.93.

# Check points

1. Compare the memory space occupied by number 3.141592 and the string "3.141592".
2. What functions can be used to convert the string " 3.141592" to a real number ?
3. What is the difference between printf() and sprintf() ?



# Array of strings

- If I have multiple strings, what is the best structure to store them in ?
  - ( Example ) " init", "open", "close", ...
- Two ways .
  - String array
  - Array of character pointers

" init "

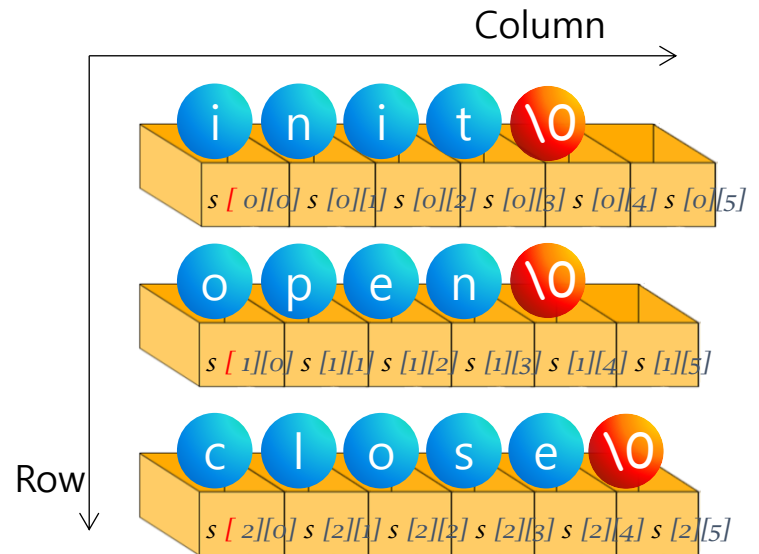
"open"

"close"



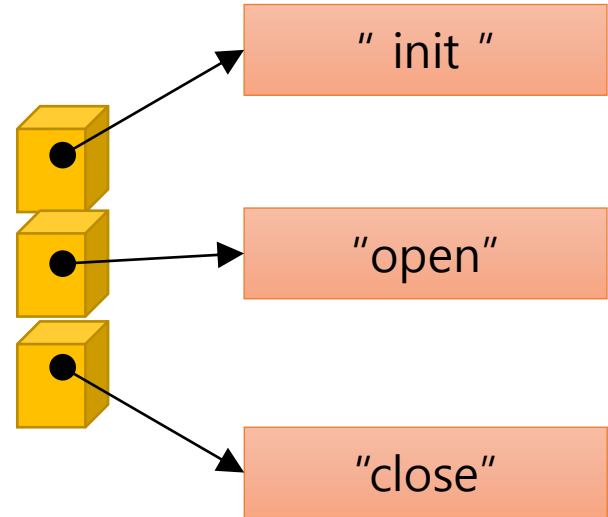
# 1. String array

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



## 2. Array of character pointers

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



# Example

```
#include <stdio.h>

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

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

    return 0;
}
```

0th menu : init  
1st menu : open  
2nd menu : close  
3rd menu : read  
4th menu : write

# Input as a two- dimensional array

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

Don't put & in front

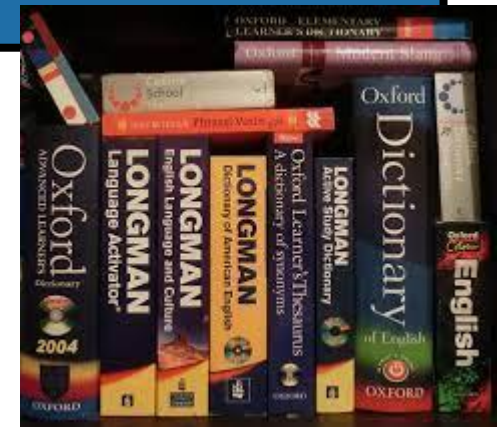


```
Enter the name of the fruit : apple
Enter the name of the fruit : Pear
Enter the name of the fruit : Grape
0th fruit : apple
1st fruit : pear
2nd fruit : grapes
```

# Lab: Korean-English Implementation of the dictionary

- Let's implement a simple Korean-English dictionary using a three- dimensional string array .

Enter the word : boy  
boy: boy



# Implementation of Korean-English dictionary

```
#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];
```

# Implementation of Korean-English dictionary

```
printf ( "Enter a word:" );  
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 ( " from dictionary Not found did not .\n" );  
  
}
```

Enter a word : book  
book: 책

# Q & A

