11. Strings

C Programming

Agenda

- Strings in C Language
- String Input/Output Functions
- String Manipulation Functions
- String/Data Conversion

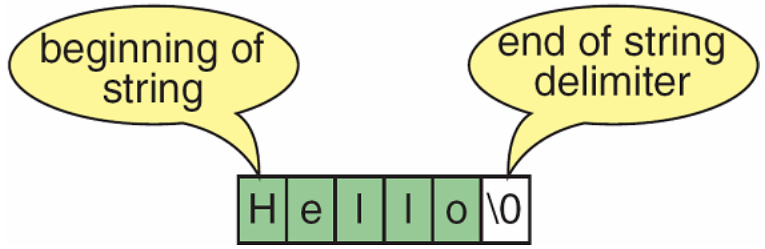
String Concepts

- String: ordered sequence of characters (or symbols)

  Ex) “Hello”, “Welcome to Handong Global University”

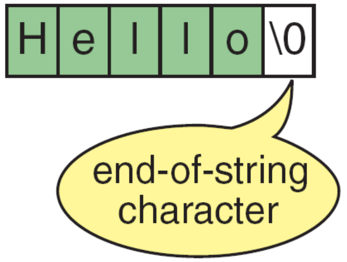
- String in C language
  - String literals are enclosed by double quotes
  - String is represented by array of character

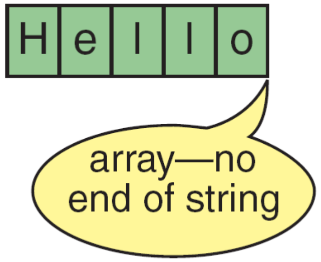
    Ex) char message[6] = “Hello”;
  - End of string is denoted by null character (‘₩0’)
    - Variable length string



String vs. Character Array

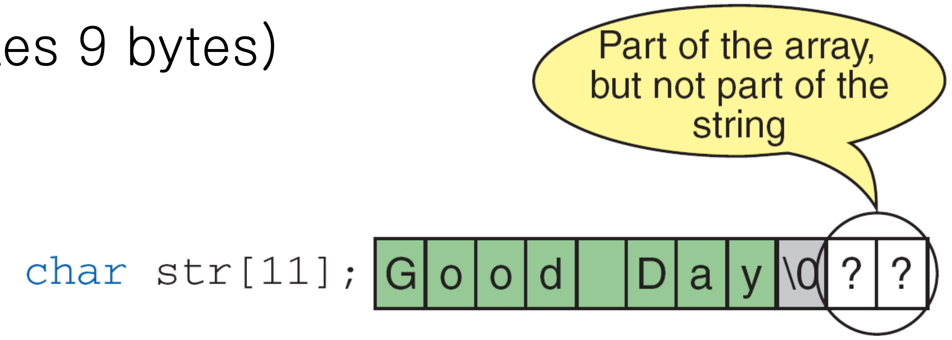
- String vs. character array
  - String terminates by null character '₩0'





- String length vs. array length

  Ex) char str[11] = "Good Day";
  - String length = 8 (takes 9 bytes)
  - Array length = 11



Declaration and Initialization

- Declaration: same with declaration of character array

  Ex) char str[9];

  - Size should be string length + 1

- Initialization
  - char string[9] = "Good Day";

    cf. char string[30] = "Good Day";
  - char string[] = "Good Day";
  - char *string = "Good Day";
  - char string[] = { 'G', 'o', 'o', 'd', ' ', 'D', 'a', 'y', '₩0' };

String and Assignment Operator

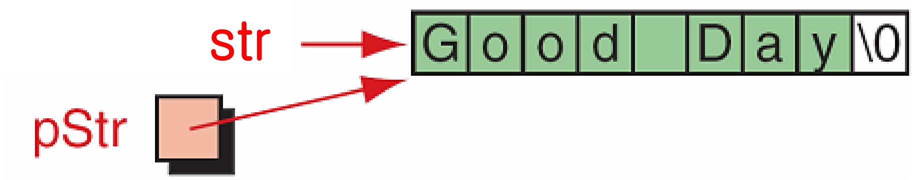
- Assigning string constants

```
char str1[9] = “Good Day”;   // Permitted only for initialization
char str2[9];
str2 = str1;                 // error!
str1 = “Good Day”;           // error!
```

  - Array assignment is not allowed in C language!

- Assigning string to a character pointer

```
char str[9] = “Good Day”;
char *pStr = NULL;
pStr = str;            // OK
```



Array of Strings

- 2D array of char type

```
int i = 0;
char aDays[7][10] = {
   “Sunday",
   “Monday",
   “Tuesday”,
   “Wednesday”,
   …
   “Saturday"
};
for(i = 0; i < 5; i++)
   printf("%s₩n", aDays[i]);
```

S	u	n	d	a	y	₩0			
M	o	n	d	a	y	₩0			
T	u	e	s	d	a	y	₩0		
W	e	d	n	e	s	d	a	y	₩0

⋮

S	a	t	u	r	d	a	y	₩0	

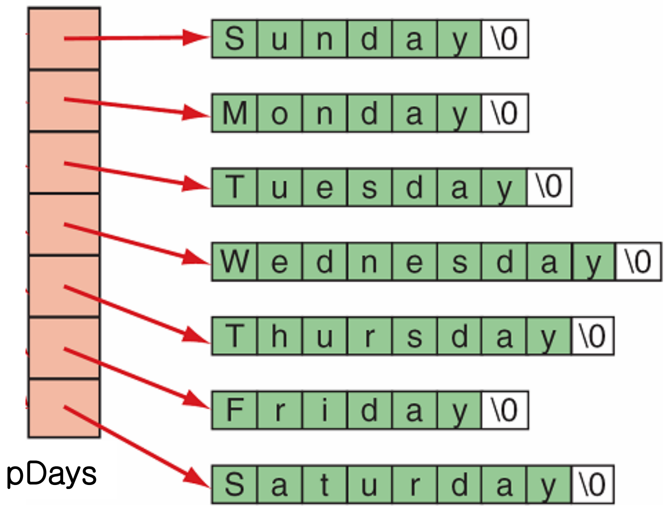
Array of Strings

- Array of pointers

```
char *pDays[7];          // array of pointer
int i = 0;               // counter variable

pDays[0] = “Sunday”;
pDays[1] = “Monday”;
…
pDays[6] = “Saturday”;

for(i = 0; i < 7; i++)
   printf(“%s₩n”, pDays[i]);
```



Agenda

- Strings in C Language
- <u>String Input/Output Functions</u>
- String Manipulation Functions
- String/Data Conversion

String I/O Functions

- Formatted string I/O: printf, scanf

  Ex) char message[256];   // array size should be sufficient

      scanf("%s", message);   // & is not necessary

      printf("Message is %s", message);

  - Delimiter of scanf: white space characters
  - For safety, it is desirable to specify width modifier

    Ex) scanf("%255s", message);

String I/O Functions

- Console string I/O
  - Input: char *gets(char* strptr);
    - strptr: buffer to store the input string
      - Delimiter: ‘₩n’, (‘₩n’ is replaced with ‘₩0’)
    - Return value
      - Success: strptr
      - Failure: NULL
  - Output: int puts(char* strptr);
    - strptr: string to print
      - ‘₩n’ is appended automatically
    - Return value
      - Success: non-negative integer
      - Failure: EOF

Example

- Declaration

```
char buffer[256];
// input is [Hello, World↵]
```

- Input with scanf

```
scanf("%s", buffer);          // buffer will store "Hello,"
```

- Input with fgets

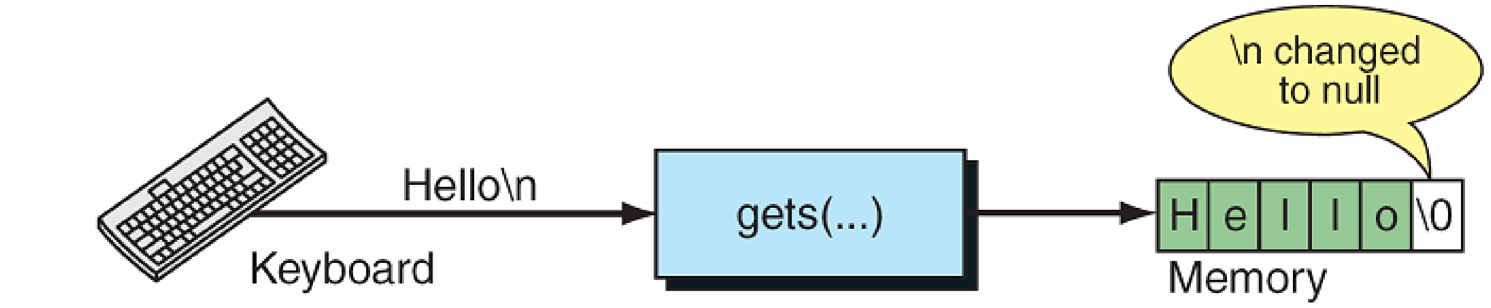
```
gets(buffer);                 // buffer will store "Hello, World"
```

String I/O Functions

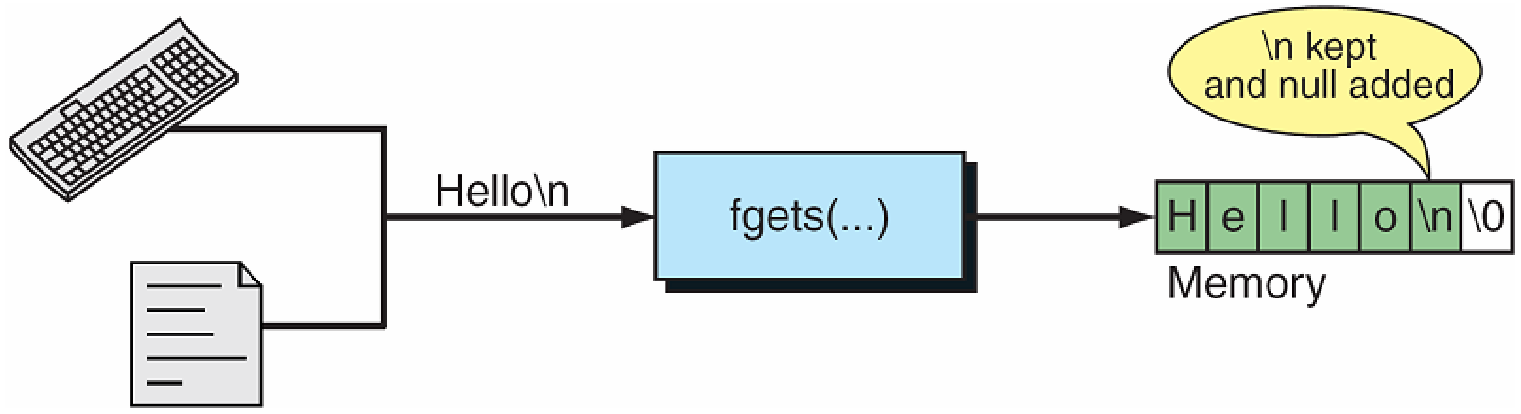
- String file I/O
  - Input: char *fgets(char *strPtr, int size, FILE *sp);
    - strPtr: buffer to read string
    - size: size of buffer
      - Maximum characters read: size – 1
    - sp: stream pointer
    - Return value
      - Success: strPtr
      - Failure: NULL
  - Output: int fputs(char *strPtr, FILE *sp);
    - Return value
      - Success: 0
      - Failure: EOF

gets() vs. fgets()

- gets replaces '₩n' with '₩0'



- fgets keeps '₩n' and appends '₩0'



Agenda

- Strings in C Language
- String Input/Output Functions
- <u>String Manipulation Functions</u>
- String/Data Conversion

String Manipulation Functions

- Given two strings str1, str2

  ```
  char str1[10] = “123”;
  char str2[10] = “456”
  char str3[10];
  ```

- Are these correct in C? NO!
  - Assignment or copy

    ```
    str3 = str1;
    ```

  - Comparison

    ```
    if(str1 == str2) …
    If(str1 < str2) …
    ```

  - Concatenation

    ```
    str3 = str1 + str2;          // Is the result “123456”?
    ```

String Manipulation Functions

- String functions (declared in string.h)
  - String length: strlen
  - String copy: strcpy, strncpy
  - String compare: strcmp, strncmp
  - String concatenation: strcat, strncat

String Length

- Syntax: int strlen(const char *string);
  - string: input string
  - Return value: length of string
    - '₩0' is not counted

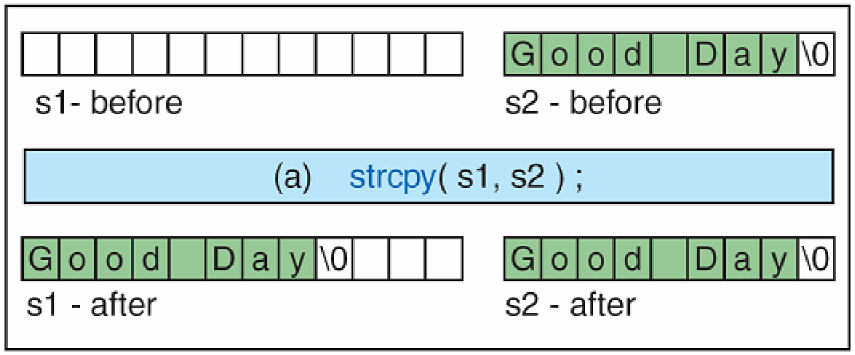
Ex)

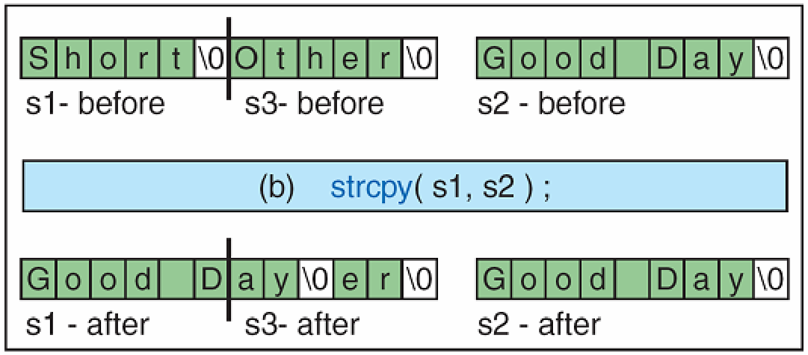
```
char *string = "Hello";
printf("length of [%s] = %d₩n", string, strlen(string));
// result: length of [Hello] = 5
```

String Copy

- Syntax
  - char* strcpy(char *toStr, const char* fromStr);
    - toStr: string buffer (destination)
    - fromStr: string to be copied (source)
    - Return value: toStr

Note! strcpy can be not safe!

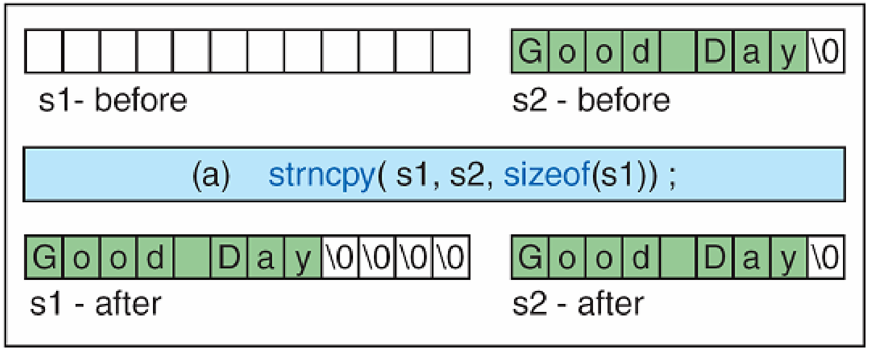


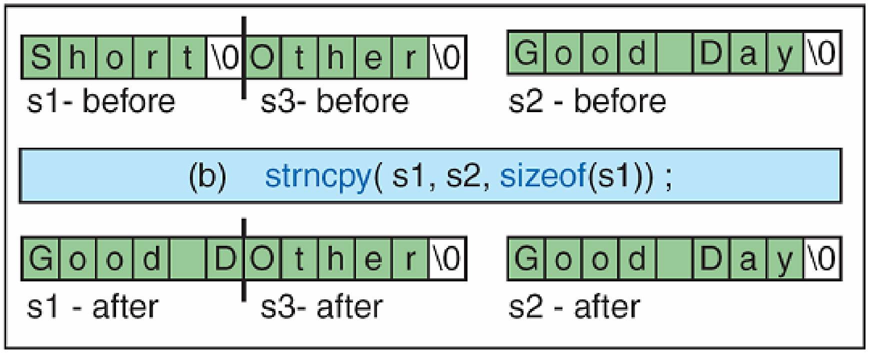


Copying long string

String Copy

- String copy with length control
  - char* strncpy(char *toStr, const char* fromStr, size_t maxLen);
    - maxLen: maximum # of characters to copy
  - Note! If maxLen is not large enough, '₩0' can be omitted!





String Compare

- String compare
  - 'less than' and 'greater than' relation of string are decided by alphabetical order

    Ex) "Hello" < "World", "abcde" > "ABCDE"

  - Syntax: `int strcmp(const char*str1, const char*str2);`
    - str1, str2: two strings to compare
    - Return value
      - 0: str1 and str2 stores the same string

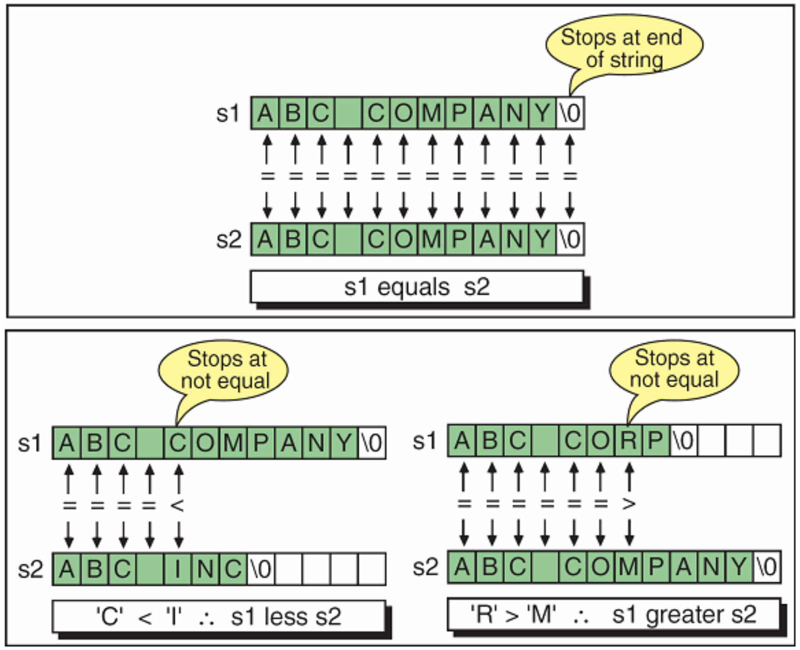
        strcmp("Hello", "Hello") == 0
      - Positive integer: str1 follows str2

        strcmp("abcde", "ABCDE") > 0
      - Negative integer: str1 precedes str2

        strcmp("Hello", "World") < 0

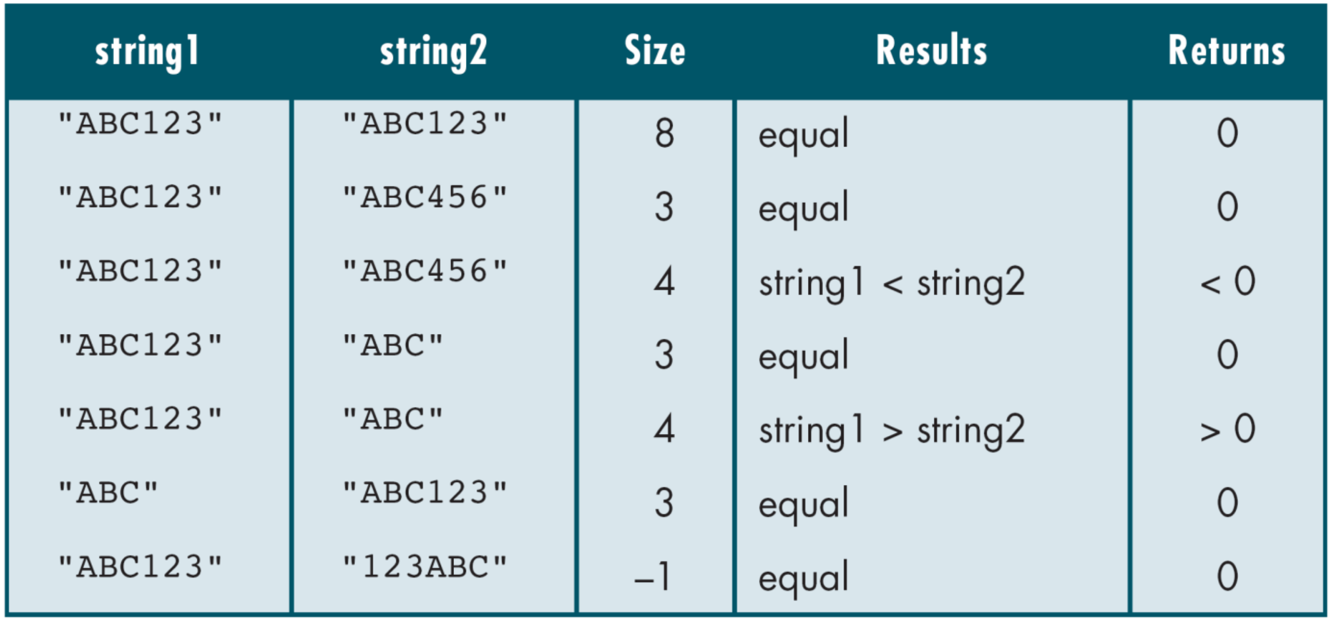
String Compare

- Behavior of strcmp
  - Compares each character at str1 with the character at the same position in str2, from left to right.
    - If a difference is found, stop comparison, return negative/positive number.
    - If '₩n' is reached, return 0



String Compare

- String compare with length limit
  - Syntax: int strncmp(const char *str1, const char *str2, int maxLen);
    - maxLen: maximum # of characters to compare



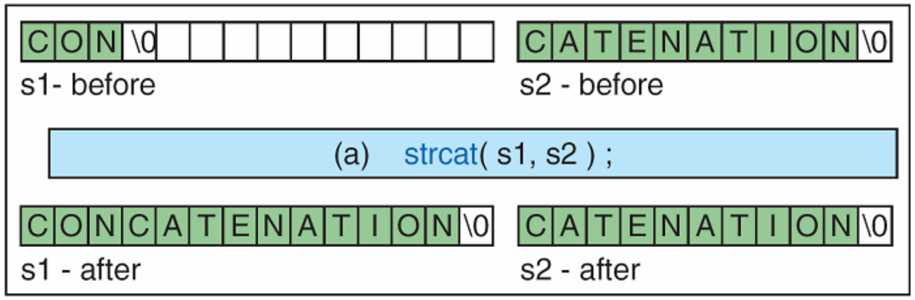
string1	string2	Size	Results	Returns
"ABC123"	"ABC123"	8	equal	0
"ABC123"	"ABC456"	3	equal	0
"ABC123"	"ABC456"	4	string1 < string2	< 0
"ABC123"	"ABC"	3	equal	0
"ABC123"	"ABC"	4	string1 > string2	> 0
"ABC"	"ABC123"	3	equal	0
"ABC123"	"123ABC"	–1	equal	0

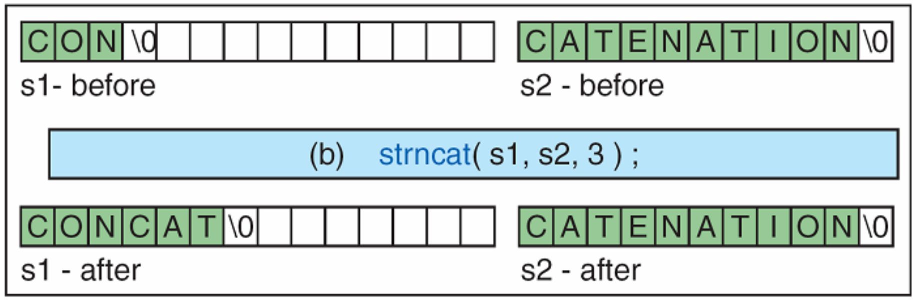
String Concatenate

- String concatenation: appending a string to the end of another string

  Ex) “con” + “catenation” = “concatenation”

  - Syntax: char* strcat(char* str1, const char* str2);
    - str2 is copied the end of str1
- Length-controlled
  - Syntax: char* strncat(char* str1, const char* str2, int maxLen);
    - maxLen: maximum # of characters to concatenate





Other String Manipulation Functions

- Search for a character
  - char* strchr(const char *string, int ch);
  - char* strrchr(const char *string, int ch);
- Search for a substring
  - char* strstr(const char* string, const char* sub_string);
- Search for character in set
  - int strspn(const char* str, const char *set);

Other String Manipulation Functions

- String to numbers

Numeric Format	ASCII Function	Wide-character Function
double	strtod	wcstod
float	strtof	wcstof
long double	strtold	wcstold
long int	strtol	wcstol
long long int	strtoll	wcstoll
unsigned long int	strtoul	wcstoul
unsigned long long int	strtoull	wcstoull

Agenda

- Strings in C Language
- String Input/Output Functions
- String Manipulation Functions
- <u>String/Data Conversion</u>

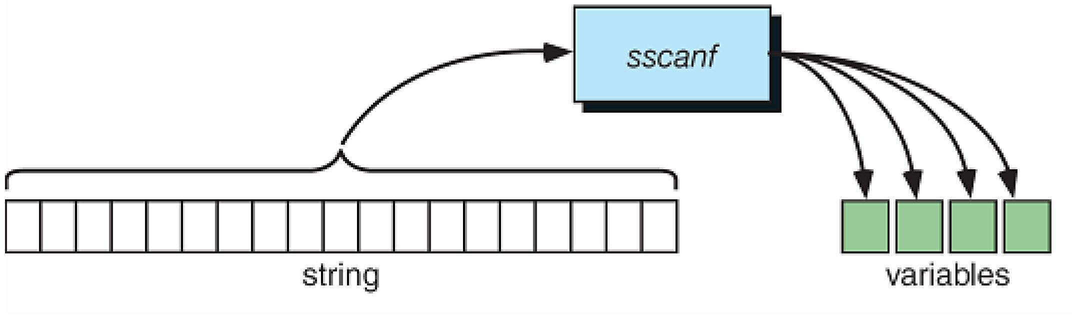
String/Data Conversion

- Stream/Data conversion
  - Conversion from stream to values: scanf, fscanf
  - Conversion from values to stream: printf, fprintf
- String/Data conversion
  - Conversion from string to values: sscanf
  - Conversion from values to string: sprintf
- String to integer/float

String/Data Conversion

- Conversion from string to values
  - Syntax: int sscanf(char *str, const char* format_string, address_list);

  Ex) sscanf("35 x 50", "%d %*c %d", &width, &height);
  - Header file: stdio.h

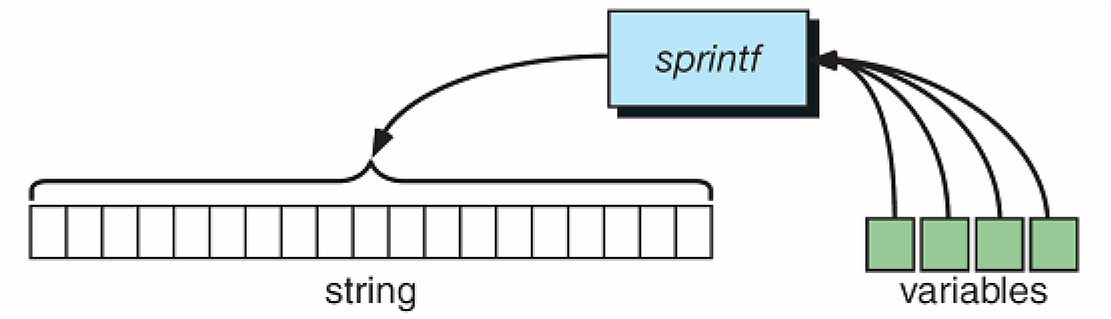


String/Data Conversion

- Conversion from values to string
  - Syntax: int sprintf(char *str, const char* format_string, value_list);

Ex) char message[128]; // array size should be large enough
sprintf(message, "width = %d, height = %d₩n", width, height);

  - Header file: stdio.h



Integer To Integer/Float

- Header file: stdlib.h
- Conversion from string to integer
  - int atoi(const char *str);
    - Return value: converted value (If conversion fails, returns 0)

```
Ex) char buffer[256];
    int value = 0;
    scanf("%255s", buffer);
    value = atoi(buffer);
```

- Conversion from string to long integer
  - long atol(const char *str);
- Conversion from string to float
  - float atof(const char *str);