



Mahidol University

ITCS113

Fundamentals of Programming

Lecture 12 - String

Instructor: Asst. Prof. Dr. Akara Supratak

Contact: akara.sup@mahidol.edu



Today's Topics

- String
 - Initializing a string variable
 - String input/output
 - String library functions
 - Loop through string



String

What is String?

- **String:** a sequence of characters enclosed in double quotes.
 - "Good Morning!"
- There is NO string type in C.
- A string is stored as an array of characters terminated with `'\0'`.
 - `char my_string[] = "Good Morning!";`



G	o	o	d		M	o	r	n	i	n	g	!	\0
---	---	---	---	--	---	---	---	---	---	---	---	---	----

Initializing a String Variable

The C compiler treats a string as an abbreviation for an array initialization.

```
char string_name[] = "Any characters ...";
```

```
char date[] = "June 14";
```

```
char date[] = {'J','u','n','e',' ','1','4','\0'};
```

J	u	n	e		1	4	\0
---	---	---	---	--	---	---	----

Initializing a String Variable

We can **specify the size** of the array at the initialization as well.

```
char string_name[size] = "Any characters ...";
```

Note: The size of the array should be **larger** than the maximum length of the string.

```
#define MAX_LEN 10  
char date1[MAX_LEN+1] = "June 14"; // +1 for '\0'  
char date2[MAX_LEN+1] = "May 14";
```

date1	J	u	n	e		1	4	\0	\0	\0	\0
date2	M	a	y		1	4	\0	\0	\0	\0	\0

Exercise

What is the difference between date1 and date2?

```
#define MAX_LEN 10  
char date1[MAX_LEN+1] = "June 14"; // +1 for '\0'  
char date2[] = "June 14";
```



String Input and Output

String Input and Output

There are several built-in C functions that we can use:

Input

`fgets()`

`scanf()`

`getchar()`

Output

`puts()`

`printf()`

`putchar()`

String Input and Output

`fgets()`

```
char *fgets(char *str, int n, FILE *stream)
```

`fgets()` reads a line from a terminal and stores it into the string pointed to by `str`.

- `str`: the pointer to an array of chars to store the input string
- `n`: the maximum number of chars to be read (including `'\0'`)
- `stream`: the pointer to a `FILE` object that identifies the stream where chars are read from

It stops when whichever below comes first:

- $(n-1)$ characters are read
- Newline (`\n`) character is read
- End-of-file (`EOF`) is reached



String Input and Output

`fgets()`

For example,

```
#include <stdio.h>

#define MAX_LEN 15

int main()
{
    char input_str[MAX_LEN];
    fgets(input_str, MAX_LEN, stdin);
    printf("%s", input_str);

    return 0;
}
```

String Input and Output

`fgets()`

For example,

```
#include <stdio.h>

#define MAX_LEN 15

int main()
{
    char input_str[MAX_LEN];
    fgets(input_str, MAX_LEN, stdin);
    printf("%s", input_str);

    return 0;
}
```

Input stream is from the terminal
(i.e., standard input).

`printf("%s", ...);` can be
used to print the string on the
terminal.



String Input and Output

fgets()

```
#include <stdio.h>

#define MAX_LEN 15

int main()
{
    char input_str[MAX_LEN];
    fgets(input_str, MAX_LEN, stdin);
    printf("%s", input_str);

    return 0;
}
```

temp.c

```
$ gcc temp.c -o temp && ./temp
Hello World!
Hello World!
$ gcc temp.c -o temp && ./temp
Fundamentals of Programming
Fundamentals o$ gcc temp.c -o temp
&& ./temp
MUICT
MUICT
$ gcc temp.c -o temp && ./temp
a_very_long_sentence
a_very_long_se$
```



String Input and Output

`fgets()`

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

temp.c

```
#define MAX_LEN 15
```

```
int main()
```

```
{
```

```
    char input_str[MAX_LEN];
```

```
    fgets(input_str, MAX_LEN,  
stdin);
```

```
    printf("%s", input_str);
```

```
    return 0;
```

```
}
```

```
$ gcc temp.c -o temp && ./temp
```

```
Hello World!
```

```
Hello World!
```

```
$ gcc temp.c -o temp && ./temp
```

```
Fundamentals of Programming
```

```
Fundamentals o$ gcc temp.c -o temp && ./temp
```

```
MUICT
```

```
MUICT
```

```
$ gcc temp.c -o temp && ./temp
```

```
a_very_long_sentence
```

```
a_very_long_se$
```

fgets() keeps '\n' if the length of the input string is *less than* MAX_LEN.



String Input and Output

fgets()

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

#define MAX_LEN 15

int main()
{
    char input_str[MAX_LEN];
    fgets(input_str, MAX_LEN, stdin);
    char *pos;
    if ((pos=strchr(input_str, '\n')) != NULL)
        *pos = '\0';
    printf("%s", input_str);

    return 0;
}
```

We can remove '\n' by adding the following codes after fgets()

```
char *pos;
if ((pos=strchr(input_str, '\n')) != NULL)
    *pos = '\0';
```

We use strchr function to search for the first occurrence of '\n', so we need to have `#include <string.h>` as well

Read more about strchr function here:

https://www.tutorialspoint.com/c_standard_library/c_function_strchr.htm



String Input and Output

fgets()

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

#define MAX_LEN 15

int main()
{
    char input_str[MAX_LEN];
    fgets(input_str, MAX_LEN, stdin);
    char *pos;
    if ((pos=strchr(input_str, '\n')) != NULL)
        *pos = '\0';
    printf("%s", input_str);

    return 0;
}
```

temp.c

'\n' was removed

```
$ gcc temp.c -o temp && ./temp
Hello World!
Hello World!$ gcc temp.c -o temp && ./temp
Fundamentals of Programming
Fundamentals o$ gcc temp.c -o temp && ./temp
MUICT
MUICT$ gcc temp.c -o temp && ./temp
a_very_long_sentence
a_very_long_se$
```


String Input and Output

`scanf("%s", str)`

- Read characters **until the first whitespace or the newline.**
- A terminating null-character ('`\0`') is automatically added at the end of the string.

```
#include <stdio.h>
#define MAX_LEN 15

int main()
{
    char input_str[MAX_LEN];
    scanf("%s", input_str);
    printf("%s", input_str);
    return 0;
}
```



String Input and Output

`scanf("%s", str)`

```
#include <stdio.h>
#define MAX_LEN 15

int main()
{
    char input_str[MAX_LEN];
    scanf("%s", input_str);
    printf("%s", input_str);
    return 0;
}
```

temp.c

```
$ gcc temp.c -o temp && ./temp
Hello World!
Hello$ gcc temp.c -o temp && ./temp
Fundamentals of Programming
Fundamentals$ gcc temp.c -o temp && ./temp
MUICT
MUICT$ gcc temp.c -o temp && ./temp
a_very_long_sentence
Abort trap: 6
$
```



String Input and Output

`scanf("%s", str)`

```
#include <stdio.h>
#define MAX_LEN 15

int main()
{
    char input_str[MAX_LEN];
    scanf("%s", input_str);
    printf("%s", input_str);
    return 0;
}
```

temp.c

```
$ gcc temp.c -o temp && ./temp
Hello World!
Hello$ gcc temp.c -o temp && ./temp
Fundamentals of Programming
Fundamentals$ gcc temp.c -o temp && ./temp
MUICT
MUICT$ gcc temp.c -o temp && ./temp
a_very_long_sentence
Abort trap: 6
$
```

Error !!



String Input and Output

getchar()

We can use `getchar()` with loop to receive a string from the terminal.

```
#include <stdio.h>
#define MAX_LEN 15

int main()
{
    char input_str[MAX_LEN];
    int i = 0;
    char c = getchar();
    while((i < MAX_LEN - 1) && (c != '\n')) {
        input_str[i++] = c;
        c = getchar();
    }
    input_str[i] = '\0';

    printf("%s", input_str);
    return 0;
}
```

temp.c

String Input and Output

getchar()

We can use `getchar()` with loop to receive a string from the terminal.

```
#include <stdio.h>
#define MAX_LEN 15

int main()
{
    char input_str[MAX_LEN];
    int i = 0;
    char c = getchar();
    while((i < MAX_LEN - 1) && (c != '\n')) {
        input_str[i++] = c;
        c = getchar();
    }
    input_str[i] = '\0';

    printf("%s", input_str);
    return 0;
}
```

temp.c

```
$ gcc temp.c -o temp && ./temp
Hello World!
Hello World!$ gcc temp.c -o temp && ./temp
Fundamental of Programming
Fundamental of$ gcc temp.c -o temp && ./temp
Fundamentals of Programming
Fundamentals o$ gcc temp.c -o temp && ./temp
MUICT
MUICT$ gcc temp.c -o temp && ./temp
a_very_long_sentence
a_very_long_se$
```



String Library Functions

String Library Functions

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

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

Name	Description
<code>strcat(string1, string2)</code>	Concatenates <code>string2</code> to <code>string1</code> .
<code>strcpy(string1, string2)</code>	Copies <code>string2</code> to <code>string1</code> .
<code>strlen(string)</code>	Returns the length of the string.
<code>strchr(string, character)</code>	Locates the position of the first occurrence of the character within the string. Returns the address of the character.
<code>strcmp(string1, string2)</code>	Compares <code>string2</code> to <code>string1</code> .
<code>isalpha(character)</code>	Returns a nonzero number if the character is a letter; otherwise it returns a zero.
<code>isupper(character)</code>	Returns a nonzero number if the character is uppercase; otherwise it returns a zero.
<code>islower(character)</code>	Returns a nonzero number if the character is lowercase; otherwise it returns a zero.
<code>isdigit(character)</code>	Returns a nonzero number if the character is a digit (0 through 9); otherwise it returns a zero.
<code>toupper(character)</code>	Returns the uppercase equivalent if the character is lowercase; otherwise it returns the character unchanged.
<code>tolower(character)</code>	Returns the lowercase equivalent if the character is uppercase; otherwise it returns the character unchanged.



String Library Functions

`strcpy()` vs. `strncpy()`

- `strcpy(strto, strfrom)`: copy strfrom to strto
- `strncpy(strto, strfrom, n)`: copy n chars from strfrom to strto

`strcmp()` vs. `strncmp()`

- `strcmp(str1, str2)`: compare str1 and str2
- `strncmp(str1, str2, n)`: compare first n chars of str1 and str2

`strcat()` vs. `strncat()`

- `strcat(strto, strfrom)`: append strfrom to strto
- `strncat(strto, strfrom, n)`: append n chars from strfrom to strto

String Library Functions

`strchr()` vs. `strrchr()`

- `strchr(str, c)`: find char `c` in `str` and return pointer to first occurrence
- `strrchr(str, c)`: find char `c` in `str` and return pointer to last occurrence

And more ...

Compare Two Strings

Compare individual characters using the decimal codes from the ASCII table.

```
strcmp(str1, str2)
```

Return

- 0: if both string are identical
- **negative**: if the ASCII value of the first unmatched character in `str1` is **less than** in `str2`.
- **positive**: if the ASCII value of the first unmatched character in `str1` is **greater than** in `str2`.

Compare Two Strings

C Code	Return	Meaning	Explanations
<code>strcmp("Good Bye", "Hello")</code>	negative	<code>"Good Bye" < "Hello"</code>	The first 'G' in Good Bye is less than the first 'H' in Hello
<code>strcmp("Hello", "Hello ")</code>	negative	<code>"Hello" < "Hello "</code>	The '\0' terminating the first string is less than the ' ' (blank) in the second string
<code>strcmp("123", "122")</code>	positive	<code>"123" > "122"</code>	'3' in 123 is greater than '2' in 122
<code>strcmp("1237", "123")</code>	positive	<code>"1237" > "123"</code>	'7' in 1237 is greater than '\0' in 123
<code>strcmp("Mahidol", "mahidol")</code>	negative	<code>"Mahidol" < "mahidol"</code>	'M' in Mahidol is less than 'm' in mahidol
<code>strcmp("MUICT", "MUICT")</code>	0	<code>"MUICT" == "MUICT"</code>	Both are identical

Compare Two Strings

Caution: CANNOT use `==`, `>=`, `<=`, `!=`, etc. to compare strings

- Because string is an array of characters

Must use `strcmp(str1, str2)`

- `strcmp(str1, str2) < 0` ~ `str1 < str2`
- `strcmp(str1, str2) > 0` ~ `str1 > str2`
- `strcmp(str1, str2) == 0` ~ `str1 == str2`
- `strcmp(str1, str2) <= 0` ~ `str1 <= str2`
- `strcmp(str1, str2) >= 0` ~ `str1 >= str2`
- ...



Example: Compare Two Strings

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

int main()
{
    char str1[] = "Google";
    char str2[] = "Microsoft";
    char str3[] = "Apple";
    int result;

    result = strcmp(str1, str2);
    if (result == 0) printf("%s == %s\n", str1, str2);
    else if (result > 0) printf("%s > %s\n", str1, str2);
    else printf("%s < %s\n", str1, str2);

    result = strcmp(str2, str3);
    if (result == 0) printf("%s == %s\n", str2, str3);
    else if (result > 0) printf("%s > %s\n", str2, str3);
    else printf("%s < %s\n", str2, str3);

    result = strcmp(str1, str3);
    if (result == 0) printf("%s == %s\n", str1, str3);
    else if (result > 0) printf("%s > %s\n", str1, str3);
    else printf("%s < %s\n", str1, str3);

    return 0;
}
```



Example: Compare Two Strings

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

int main()
{
    char str1[] = "Google";
    char str2[] = "Microsoft";
    char str3[] = "Apple";
    int result;

    result = strcmp(str1, str2);
    if (result == 0) printf("%s == %s\n", str1, str2);
    else if (result > 0) printf("%s > %s\n", str1, str2);
    else printf("%s < %s\n", str1, str2);

    result = strcmp(str2, str3);
    if (result == 0) printf("%s == %s\n", str2, str3);
    else if (result > 0) printf("%s > %s\n", str2, str3);
    else printf("%s < %s\n", str2, str3);

    result = strcmp(str1, str3);
    if (result == 0) printf("%s == %s\n", str1, str3);
    else if (result > 0) printf("%s > %s\n", str1, str3);
    else printf("%s < %s\n", str1, str3);

    return 0;
}
```

Google < Microsoft
Microsoft > Apple
Google > Apple

As you can see, we have several lines of code that do the same/similar things over and over in this program.
How can we simplify this code?

Example: Compare Two Strings

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

void compare_string(char *str1, char *str2)
{
    int result = strcmp(str1, str2);
    if (result == 0) printf("%s == %s\n", str1, str2);
    else if (result > 0) printf("%s > %s\n", str1, str2);
    else printf("%s < %s\n", str1, str2);
}

int main()
{
    char str1[] = "Google";
    char str2[] = "Microsoft";
    char str3[] = "Apple";

    compare_string(str1, str2);
    compare_string(str2, str3);
    compare_string(str1, str3);

    return 0;
}
```

```
Google < Microsoft
Microsoft > Apple
Google > Apple
```

As you can see, we have several lines of code that do the same/similar things over and over in this program.
How can we simplify this code?
Using function!



Example: Assign String Value

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

#define MAX_LEN 15

int main()
{
    char str1[MAX_LEN] = "Google"; // if no MAX_LEN, we cannot store a longer string.
    char str2[MAX_LEN] = "Microsoft";

    // str2 = str1; Error !! --> Cannot assign the value with '='
    strcpy(str2, str1); // strcpy() is typically used to assign string value
    printf("%s %s\n", str1, str2);

    return 0;
}
```




Example: Assign String Value

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

int main()
{
    char str1[] = "Google";
    char str2[] = "Microsoft";
    char str3[] = "Apple";

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

    strcpy(str3, str1); // Error !! --> the size of str3 is smaller than str1
    printf("%s %s\n", str1, str3);

    return 0;
}
```

Ref: <https://linux.die.net/man/3/strcpy>



Example: Concat Two Strings

```
#include <stdio.h>
#include <string.h>
#define MAX_LEN 20

int main() {
    char fname[MAX_LEN], lname[MAX_LEN], fullname[MAX_LEN];
    scanf("%s", fname);
    scanf("%s", lname);
    printf("# of chars in fname: %lu\n", strlen(fname));
    printf("# of chars in lname: %lu\n", strlen(lname));

    strcpy(fullname, fname);
    strcat(fullname, " ");
    strcat(fullname, lname);

    printf("Fullname: %s\n", fullname);
    printf("# of chars: %lu\n", strlen(fullname));
    return 0;
}
```

Example: Concat Two Strings

```
#include <stdio.h>
#include <string.h>
#define MAX_LEN 20

int main() {
    char fname[MAX_LEN], lname[MAX_LEN], fullname[MAX_LEN];
    scanf("%s", fname);
    scanf("%s", lname);
    printf("# of chars in fname: %lu\n", strlen(fname));
    printf("# of chars in lname: %lu\n", strlen(lname));

    strcpy(fullname, fname);
    strcat(fullname, " ");
    strcat(fullname, lname);

    printf("Fullname: %s\n", fullname);
    printf("# of chars: %lu\n", strlen(fullname));
    return 0;
}
```

```
Akara
Supratak
# of chars in fname: 5
# of chars in lname: 8
Fullname: Akara Supratak
# of chars: 14
```



L12-ClassEx-1

Write a program to accept two strings and determine whether they are the same. If so, print 1; otherwise, print 0.

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

int main()
{
    // Read in two strings
    // Compare two strings
    // Print the output

    return 0;
}
```



Loop through String

Loop through String

Typically we use a **while-loop** as we do not know the length of the string stored in the array array.

```
char str[] = "...";
int i = 0;
...

// Check for end-of-string
while (str[i] != '\0') {
    ...
    i++;
}
...
```



Loop through String

Typically we use a **while-loop** as we do not know the length of the string stored in the array array.

```
char str[] = "This is an Alphabet.";
int i = 0;
int num_a = 0;
// Check for end-of-string
while (str[i] != '\0') {
    if (str[i] == 'A' || str[i] == 'a')
    {
        num_a++;
    }
    i++;
}
printf("%d", num_a);
```

Loop through String

Typically we use a **while-loop** as we do not know the length of the string stored in the array array.

```
char str[] = "This is an Alphabet.";
int i = 0;
int num_a = 0;
// Check for end-of-string
while (str[i] != '\0') {
    if (str[i] == 'A' || str[i] == 'a')
    {
        num_a++;
    }
    i++;
}
printf("%d", num_a);
```

If you want to use a **for-loop**,
you can use **strlen** function
to determine the string length.

J u n e \0 \0 \0

└──────────┘

Length of String

Not include \0

Loop through String

Typically we use a **while-loop** as we do not know the length of the string stored in the array array.

```
char str[] = "This is an Alphabet.";
int i = 0;
int num_a = 0;
// Check for end-of-string
while (str[i] != '\0') {
    if (str[i] == 'A' || str[i] == 'a')
    {
        num_a++;
    }
    i++;
}
printf("%d", num_a);
```

If you want to use a **for-loop**,
you can use **strlen** function
to determine the string length.

```
int length = strlen(str);
printf("Length of str: %d", length);
```

Length of str: 20



Example: Copy a String

```
#include <stdio.h>
#define MAX_LEN 20

int main() {
    char str1[] = "This_is_a_very_long_string.";
    char str2[MAX_LEN];
    int i=0;
    // Check for end-of-string and array size
    while ((str1[i] != '\0') && (i < MAX_LEN-1)) {
        str2[i] = str1[i];
        i++;
    }
    str2[i] = '\0'; // Don't forget the end-of-string
    printf("%s\n", str2);
    return 0;
}
```

This_is_a_very_long



Example: Replace Characters

```
#include <stdio.h>

int main() {
    char str[] = "This_is_a_VeRy_Long_stRIng.";
    int i = 0;
    printf("Before: %s\n", str);
    while (str[i] != '\0') {
        if (str[i] == '_') {
            str[i] = ' ';
        }
        i++;
    }
    printf("After: %s\n", str);
    return 0;
}
```

```
Before: This_is_a_VeRy_Long_stRIng.
After: This is a VeRy Long stRIng.
```



Example: Toggle Characters

```
#include <stdio.h>

int main() {
    char str[] = "This_is_a_VeRy_Long_stRIng.";

    return 0;
}
```

Before:

This_is_a_VeRy_Long_stRIng.

After: tHIS IS A veRY LONG StRING



Example: Toggle Characters

```
#include <stdio.h>

int main() {
    char str[] = "This_is_a_VeRy_Long_stRIng.";
    int i = 0;

    while (str[i] != '\0') {

        i++;
    }

    return 0;
}
```

Before:

This_is_a_VeRy_Long_stRIng.

After: tHIS IS A veRY lONG sTrIng



Example: Toggle Characters

```
#include <stdio.h>

int main() {
    char str[] = "This_is_a_VeRy_Long_stRIng.";
    int i = 0;

    while (str[i] != '\0') {
        if (isalpha(str[i])) {

        }
        i++;
    }

    return 0;
}
```

Before:

This_is_a_VeRy_Long_stRIng.

After: tHIS IS A veRY lONG StRING

Example: Toggle Characters

```
#include <stdio.h>

int main() {
    char str[] = "This_is_a_VeRy_Long_stRIng.";
    int i = 0;

    while (str[i] != '\0') {
        if (isalpha(str[i])) {
            if (islower(str[i])) {
                str[i] = toupper(str[i]);
            } else {
                str[i] = tolower(str[i]);
            }
        }
        i++;
    }

    return 0;
}
```

Before:

This_is_a_VeRy_Long_stRIng.

After: tHIS IS A veRY lONG StRING

Example: Toggle Characters

```
#include <stdio.h>
int main() {
    char str[] = "This_is_a_VeRy_Long_stRIng.";
    int i = 0;
    printf("Before: %s\n", str);
    while (str[i] != '\0') {
        if (isalpha(str[i])) {
            if (islower(str[i])) {
                str[i] = toupper(str[i]);
            } else {
                str[i] = tolower(str[i]);
            }
        }
        i++;
    }
    printf("After: %s\n", str);
    return 0;
}
```

Before:

This_is_a_VeRy_Long_stRIng.

After: tHIS IS A veRY lONG sTRIng

Example: Toggle Characters

```
#include <stdio.h>
int main() {
    char str[] = "This_is_a_VeRy_Long_stRIng.";
    int i = 0;
    printf("Before: %s\n", str);
    while (str[i] != '\0') {
        if (isalpha(str[i])) {
            if (islower(str[i])) {
                str[i] = toupper(str[i]);
            } else {
                str[i] = tolower(str[i]);
            }
        }
        i++;
    }
    printf("After: %s\n", str);
    return 0;
}
```

Before:

This_is_a_VeRy_Long_stRIng.

After: tHIS IS A veRY lONG stRING