

# CEng 230 Introduction to C Programming

Seyyit Alper SERT

Department of Computer Engineering  
2017-2018 Fall

Web Pages

Official Course Page: [ceng230.ceng.metu.edu.tr](http://ceng230.ceng.metu.edu.tr)

Learning Management System (LMS): [odtuclass.metu.edu.tr](http://odtuclass.metu.edu.tr)

Contact: [alper.sert@ceng.metu.edu.tr](mailto:alper.sert@ceng.metu.edu.tr)

# Strings

Basics

Initialization

strcpy, strncpy,

strcat, strncat,

strcmp, strncmp functions

## Declaring and Initializing String Variables

As we mentioned earlier, a string in C is implemented as an array, so declaring a string variable is the same as declaring an array of type `char`. In

```
char string_var[30];
```

the variable `string_var` will hold strings from 0 to 29 characters long. It is C's handling of this varying length characteristic that distinguishes the string data structure from other arrays. C permits initialization of string variables using a string constant as shown in the following declaration of `str`.

```
char str[20] = "Initial value";
```

Let's look at `str` in memory after this declaration with initialization.

[0]				[4]					[9]					[14]					[19]
I	n	i	t	i	a	l			v	a	l	u	e	\0	?	?	?	?	?

Notice that `str[13]` contains the character `'\0'`, the **null character** that marks the end of a string. Using this marker allows the string's length within the character array to vary from 0 to one less than the array's declared size. All of C's string-handling functions simply ignore whatever is stored in the cells following the null character. The following diagram shows `str` holding a string that is the longest it can represent—19 characters plus the null character.

[0]					[4]					[9]					[14]				[19]
n	u	m	b	e	r	s			a	n	d		s	t	r	i	n	g	s

```
#include <stdio.h>

int main( void )
{
    char a[]="abc";
    char b[]={'a','b','c','\0'};
    char c[]={'a','b','c',0};
    char d[]={'a','b','c'};
    printf("%d %d %d\n",strlen(a),strlen(b),strlen(c));
    printf("%d\n",strlen(strcpy(d,"0")));
    printf("%s\n",d);

    system("pause");
    return 0; /* indicates successful termination */
} /* end main */
```

## Arrays of Strings

Because one string is an array of characters, an array of strings is a two-dimensional array of characters in which each row is one string. The following are statements to declare an array to store up to 30 names, each of which is less than 25 characters long.

```
#define NUM_PEOPLE 30
#define NAME_LEN 25
. . .
char names[NUM_PEOPLE][NAME_LEN];
```

We can initialize an array of strings at declaration in the following manner:

```
char month[12][10] = {"January", "February", "March", "April",
                      "May", "June", "July", "August",
                      "September", "October", "November",
                      "December"};
```



```
strncpy(one_str, "A very long test string", 20);
```

[0]				[4]					[9]					[14]					[19]	
A		v	e	r	y			l	o	n	g		t	e	s	t		s	t	r

Notice that although this call to `strncpy` has prevented overflow of destination string `one_str`, it has not stored a valid string in `one_str`: There is no terminating `'\0'`. In general, one can assign as much as will fit of a source string (`source`) to a destination (`dest`) of length `dest_len` by using these two statements:

```
strncpy(dest, source, dest_len - 1);
dest[dest_len - 1] = '\0';
```

```
char one_str[20];  
one_str = "Test string";    /* Does not work */
```

**TABLE 8.1** Some String Library Functions from `string.h`

Function	Purpose: Example	Parameters	Result Type														
strcpy	Makes a copy of <b>source</b> , a string, in the character array accessed by <b>dest</b> : <code>strcpy(s1, "hello");</code>	<code>char *dest</code> <code>const char *source</code>	<code>char *</code> <table><tr><td>h</td><td>e</td><td>l</td><td>l</td><td>o</td><td>\0</td><td>?</td><td>?</td><td>...</td></tr></table>	h	e	l	l	o	\0	?	?	...					
h	e	l	l	o	\0	?	?	...									
strncpy	Makes a copy of up to <b>n</b> characters from <b>source</b> in <b>dest</b> : <code>strncpy(s2, "inevitable", 5)</code> stores the first five characters of the source in <b>s1</b> and does NOT add a null character.	<code>char *dest</code> <code>const char *source</code> <code>size_t n</code>	<code>char *</code> <table><tr><td>i</td><td>n</td><td>e</td><td>v</td><td>i</td><td>?</td><td>?</td><td>...</td></tr></table>	i	n	e	v	i	?	?	...						
i	n	e	v	i	?	?	...										
strcat	Appends <b>source</b> to the end of <b>dest</b> : <code>strcat(s1, "and more");</code>	<code>char *dest</code> <code>const char *source</code>	<code>char *</code> <table><tr><td>h</td><td>e</td><td>l</td><td>l</td><td>o</td><td>a</td><td>n</td><td>d</td><td></td><td>m</td><td>o</td><td>r</td><td>e</td><td>\0</td></tr></table>	h	e	l	l	o	a	n	d		m	o	r	e	\0
h	e	l	l	o	a	n	d		m	o	r	e	\0				
strncat	Appends up to <b>n</b> characters of <b>source</b> to the end of <b>dest</b> , adding the null character if necessary: <code>strncat(s1, "and more", 5);</code>	<code>char *dest</code> <code>const char *source</code> <code>size_t n</code>	<code>char *</code> <table><tr><td>h</td><td>e</td><td>l</td><td>l</td><td>o</td><td>a</td><td>n</td><td>d</td><td></td><td>m</td><td>\0</td><td>?</td></tr></table>	h	e	l	l	o	a	n	d		m	\0	?		
h	e	l	l	o	a	n	d		m	\0	?						
strcmp	Compares <b>s1</b> and <b>s2</b> alphabetically; returns a negative value if <b>s1</b> should precede <b>s2</b> , a zero if the strings are equal, and a positive value if <b>s2</b> should precede <b>s1</b> in an alphabetized list: <code>if (strcmp(name1, name2) == 0)...</code>	<code>const char *s1</code> <code>const char *s2</code>	<code>int</code>														
strncmp	Compares the first <b>n</b> characters of <b>s1</b> and <b>s2</b> returning positive, zero, and negative values as does <b>strcmp</b> : <code>if (strncmp(n1, n2, 12) == 0)...</code>	<code>const char *s1</code> <code>const char *s2</code> <code>size_t n</code>	<code>int</code>														
strlen	Returns the number of characters in <b>s</b> , not counting the terminating null: <code>strlen("What")</code> returns 4.	<code>const char *s</code>	<code>size_t</code>														



## Using Character Arrays to Store and Manipulate Strings

```
char string1[] = "first";
```

```
char string1[] = { 'f', 'i', 'r', 's', 't', '\0' };
```

```
/* Fig. 6.10: fig06_10.c
   Treating character arrays as strings */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
    char string1[ 20 ]; /* reserves 20 characters */
    char string2[] = "string literal"; /* reserves 15 characters */
    int i; /* counter */

    /* read string from user into array string1 */
    printf("Enter a string: ");
    scanf( "%s", string1 ); /* input ended by whitespace character */

    /* output strings */
    printf( "string1 is: %s\nstring2 is: %s\n"
           "string1 with spaces between characters is:\n",
           string1, string2 );

    /* output characters until null character is reached */
    for ( i = 0; string1[ i ] != '\0'; i++ ) {
        printf( "%c ", string1[ i ] );
    } /* end for */

    printf( "\n" );
    system("pause");
    return 0; /* indicates successful termination */
} /* end main */
```

35. What would be the output of the following code segment?

```
char myStr[100];  
printf("Please enter a string:\n");  
scanf("%s", myStr);  
printf("%d", strlen(myStr));
```

if the user has entered:

H	e	l	l	o		W	o	r	l	d	!
---	---	---	---	---	--	---	---	---	---	---	---

as input (each character entered is displayed in a box).

a) 5

b) 6

c) 11

d) 12

e) 13

q1

48) What is the output?

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

```
void main() {
```

```
    char a[]="abc";
```

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

```
    char c[]={'a','b','c',0};
```

```
    char d[]={'a','b','c'};
```

```
    printf("%d %d %d\n",strlen(a),strlen(b),strlen(c));
```

```
    printf("%d\n",strlen(strcpy(d,"\\0")));
```

```
    printf("%s\n",d);
```

```
}
```

a) 3 3 3

0

b) 3 4 4

4

abc

c) 4 4 4

4

abc

d) 3 3 4

3

abc

e) 3 4 3

4

abc

q2

49) What is the output?

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

```
void main () {
```

```
    char e[10],f[10];
```

```
    e[0]='a';e[1]='b';e[2]='\0';
```

```
    strcat(e,"c");
```

```
    printf("%d\n",strlen(e));
```

```
    printf("%s\n",e);
```

```
    strcpy(f,e);
```

```
    f[2]='d';
```

```
    printf("%d\n",strlen(f));
```

```
    printf("%s\n",f);
```

```
}
```

```
3
abc
```

```
3
```

```
abd
```

```
Devam etmek için bir tuşa basın . . .
```

q3

**47) What is the output of the below code segment?**

```
char str1[20] = "Hello";  
char str2[20] = "World!";  
strcat(str1, str2);  
printf("%d", strlen(str1));
```

a) 1

b) 6

c) 5

d) 10

e) 11

q5

50) What is the output of the following code segment?

```
char d[] = "need a hero";  
printf("%d", strchr(d, 'a') - d);
```

a) -4

b) 11

c) 6

d) 4

e) 5

42. What will be the output of the following code segment?

q6

```
char st1[ ] = "can you hear the voice?";  
char st2[ ] = "we must call the police!";  
strcpy(st2+8, st1+8);  
if(strncmp(st1+7, st2+7, 8) == 0) printf("%s", st2);  
else printf("%s", st1);
```

a) "can you call the police!"

c) "we must call the police!"

e) "we must hear the voice?"

b) "can you hear the police!"

d) "can you hear the voice?"

50) What is the output?

```
#include <stdio.h>
void main () {
    char e[10],f[10];
    strcpy(e,"abc");
    strcpy(f,"abd");
    if (strcmp(e,f))
        printf("%s\n",e);
    else
        printf("%s\n",f);
    f[3]='x';f[4]='\0';
    printf("%s\n",strchr(f,'d'));
    printf("%s\n",strstr(f,"d"));
}
```

a) abc

d

d

b) abd

dx

dx

c) abc

dx

dx

d) c

d

d

e) c

x

x

q7

**43) What would be the output after implementation of the code?**

```
char c[ ]= "a long string";  
char s[20]= "It is my";  
strncat(s,c+1,12);  
printf("%s",s);
```

- a) "It is my long strin"      b) "It is my long string"  
c) Wrong output will be produced (missing endpoint of s)  
d) Error (out of boundry)      e) None of these.