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

Learning Management System (LMS): odtuclass.metu.edu.tr

Contact: 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];
```

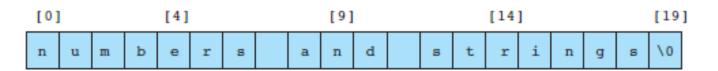
the variable string_var will hold strings from 0 to 29 characters long. It is C's handling of this varying length characteristic that distinguishes the string data structure from other arrays. C permits initialization of string variables using a string constant as shown in the following declaration of 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	1	v	a	1	u	е	\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.



```
#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:

String Assignment

Function strcpy copies the string that is its second argument into its first argument. To carry out the desired assignment shown in our faulty code above, we would write

```
strcpy(one_str, "Test String");
```

Like a call to scanf with a %s placeholder, a call to strcpy can easily overflow the space allocated for the destination variable (one_str in the example given). Variable one_str has room for up to 19 characters plus the null character. This call to strcpy

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

would overflow one_str, storing the final characters 'i', 'n', 'g', and '\0' in memory allocated for other variables. The values of these other variables would seem to change spontaneously. On rare occasions, such overflow would generate a run-time error message.

The string library provides another string-copying function named strncpy that takes an argument specifying the number of characters to copy (call this number n). If the string to be copied (the source string) is shorter than n characters, the remaining characters stored are null. For example,

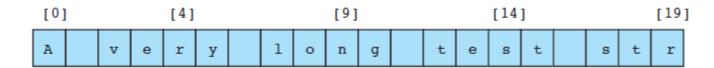
```
strncpy(one_str, "Test string", 20);
would give one_str the value:
```

[0]		[4]					[9]				[14]						[19]			
Т	е	s	t		S	t	r	i	n	g	\0	\0	\0	\0	\0	\0	\0	\0	\0	

The net effect is the same as the call

```
strcpy(one_str, "Test string");
```

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



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 source, a string, in the character array accessed by dest: strcpy(s1, "hello");	char *dest const char *source	char * h e 1 1 0 \0 ? ?
strncpy	Makes a copy of up to n characters from source in dest: strncpy(s2, "inevitable", 5) stores the first five characters of the source in s1 and does NOT add a null character.	char *dest const char *source size_t [†] n	char * i n e v i ? ?
strcat	Appends source to the end of dest: strcat(s1, "and more");	char *dest const char *source	char * h e l l o a n d m o r e \0
strncat	Appends up to n characters of source to the end of dest, adding the null character if necessary: strncat(s1, "and more", 5);	char *dest const char *source size_t [†] n	char * h e 1 1 o a n d m \0 ?
strcmp	Compares s1 and s2 alphabetically; returns a negative value if s1 should precede s2, a zero if the strings are equal, and a positive value if s2 should precede s1 in an alphabetized list: if (strcmp(name1, name2) == 0)	const char *s1 const char *s2	int
strncmp	Compares the first n characters of s1 and s2 returning positive, zero, and negative values as does strcmp: if (strncmp(n1, n2, 12) == 0)	const char *s1 const char *s2 size_t [†] n	int
strlen	Returns the number of characters in s, not counting the terminating null: strlen("What") returns 4.	const char *s	size_t

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:

Η	e	1	1	0	W	0	r	1	d	!

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

- a) 5
- b) 6 c) 11
- e) 13

```
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'};
q1
             printf("%d %d %d\n", strlen(a), strlen(b), strlen(c));
                printf("%d\n", strlen(strcpy(d, "\0")));
                printf("%s\n",d);
             a) 333
                        b) 344 c) 444 d) 334
                                                          e) 343
                                                            4
                           abc
                                      abc
                                                 abc
                                                            abc
             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");
q2
                printf("%d\n", strlen(e));
                printf("%s\n",e);
                strcpy(f,e);
                f[2]='d';
                                               3
abc
                printf("%d\n", strlen(f));
                printf("%s\n",f);
                                               abd
             }
                                               Devam etmek için bir tuşa basın .
```

```
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?
```

a) -4

- **b**) 11
- c) 6

d) 4



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

```
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);
```

q6

- 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))
q7
             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"));
                                   c) abc
            a) abc
                       b) abd
                                               d) c
                                                           e) c
              d
                          dx
                                     dx
                                                 d
                                                            \mathbf{x}
              d
                                                 d
                          dx
                                     dx
                                                            Х
```

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.