Strings
Chapter 8

Problem Solving & Program Design in C

Eighth Edition

Jeri R. Hanly & Elliot B. Koffman

Chapter Objectives

- To understand how a string constant is stored in an array of characters

- To learn about the placeholder %s and how it is used in printf and scanf operations

- To learn some of the operations that can be performed on strings such as copying strings extracting substrings, and joining strings using functions from the library string

String Basics

- A blank in a string is a valid character.
- null character
  - character '\0' that marks the end of a string in C
- A string in C is implemented as an array.
  - char string_var[30];
  - char str[20] = "Initial value";
- An array of strings is a 2-dimensional array of characters in which each row is a string.

Input/Output

- printf and scanf can handle string arguments
- use %s as the placeholder in the format string

char president[20];

scanf("%s\n", president);

printf("%s\n", president);

Initializing Strings

- sizeof() gives size in bytes
- strlen() gives length of string

char string[16] = "hello world";

char *str = "hello world";

char s[] = "hello world";

Initializing Strings

- sizeof() gives size in bytes
- strlen() gives length of string

char string[16] = "hello world";

| h | e | l | l | o | | w | o | r | l | d | \0 | | | | |

char *str = "hello world";

| 0x7ffc48aef660 | → | h | e | l | l | o | | w | o | r | l | d | \0 |

char s[] = "hello world";

| h | e | l | l | o | | w | o | r | l | d | \0 |

Initializing Strings

- sizeof() gives size in bytes
- strlen() gives length of string

char string[16] = "hello world";

sizeof() is 16
strlen() is 11

| h | e | l | l | o | | w | o | r | l | d | \0 | | | | |

char *str = "hello world";

sizeof() is 8
strlen() is 11

| 0x7ffc48aef660 | → | h | e | l | l | o | | w | o | r | l | d | \0 |

char s[] = "hello world";

| h | e | l | l | o | | w | o | r | l | d | \0 |

sizeof() is 12
strlen() is 11

Buffer Overflow

- more data is stored in an array than its declared size allows

- a very dangerous condition

- unlikely to be flagged as an error by either the compiler or the run-time system

char string[8] = "hello world";

| h | e | l | l | o | | w | o | r | l | d | \0 |

String Assignment

- strcpy
  - copies string in second argument into its first argument
    - strcpy(s1, "hello");
  - subject to buffer overflow
- strncpy
  - takes an argument specifying the number of chars to copy
  - if the string to be copies is shorter, the remaining characters stored are null
    - strncpy(s2, "inevitable", 5);

= does not work!

String tokenization

```
char *last, *first, *middle;
char pres[20] = "Adams, John Quincy";
char pres_copy[20];
strcpy(pres_copy, pres);
```
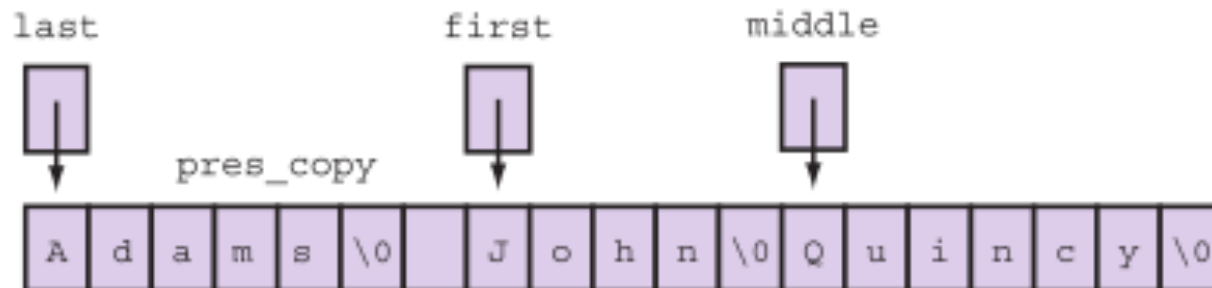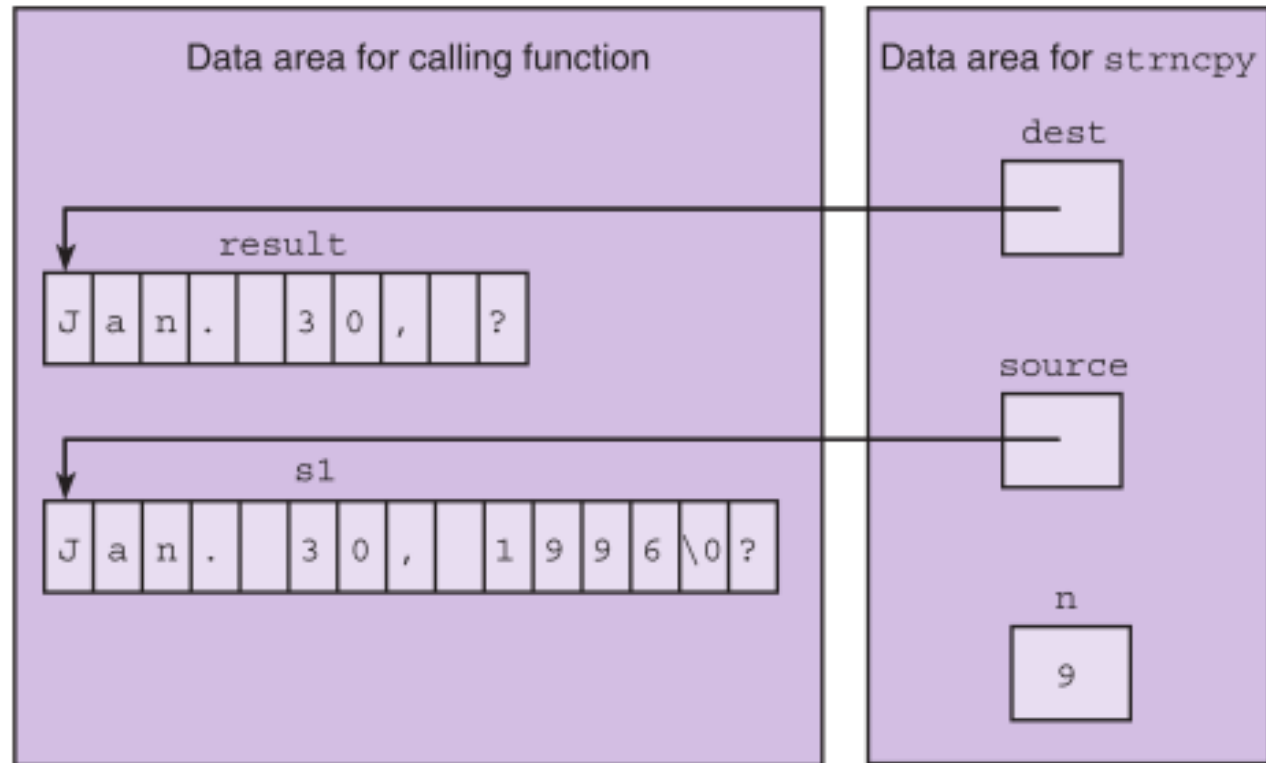


```
last = strtok(pres_copy, ", ");
first = strtok(NULL, ", ");
middle = strtok(NULL, ", ");
```
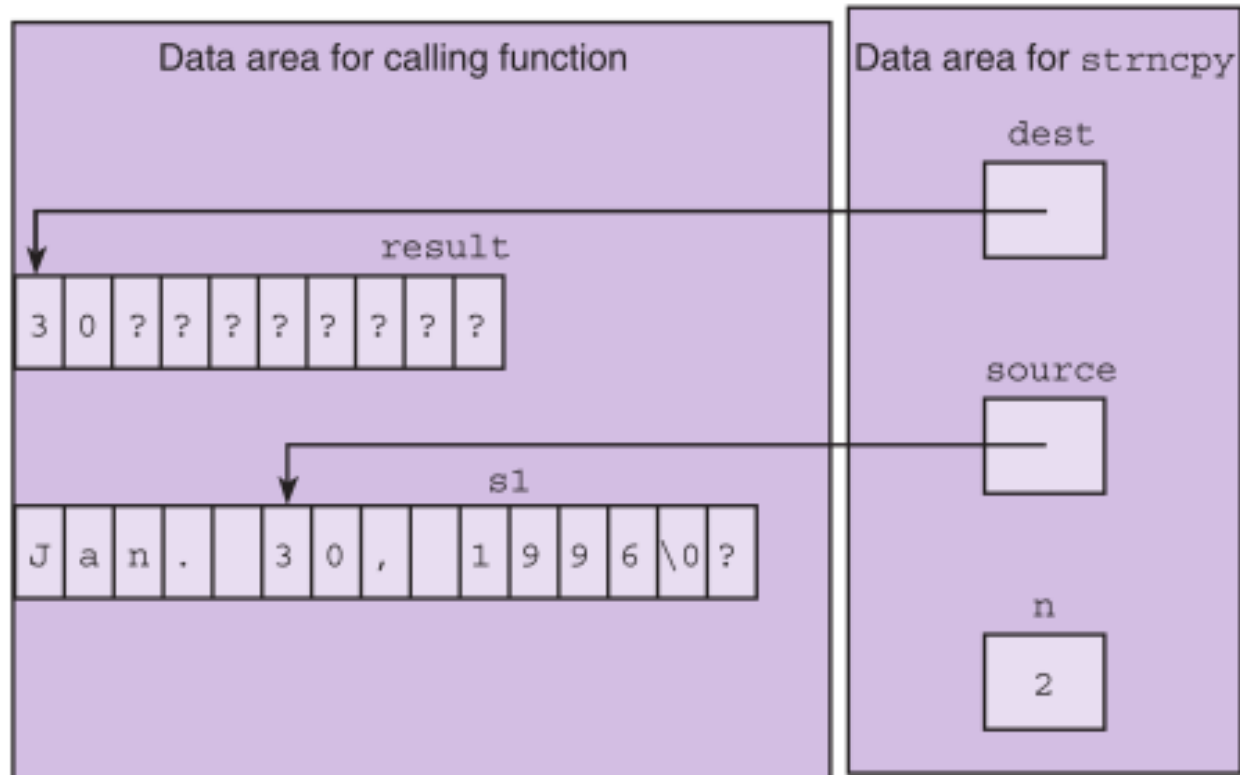
Substrings

FIGURE 8.5

Execution of
`strncpy(result, s1, 9);`

Substrings

FIGURE 8.6

Execution of
`strncpy(result,
&s1[5], 2);`

Substrings

```
char last [20], first [20], middle [20];
char pres[20] = " Adams , John Quincy ";
```

```
strncpy (last, pres, 5);
last[5] = '\0';
```

```
strcpy (middle, &pres[12]);
```

```
strncpy (first, &pres[7], 4);
first[4] = '\0';
```

Substrings

```
char last [20], first [20], middle [20];
char pres[20] = " Adams , John Quincy ";
```

```
strncpy (last, pres, 5);
last[5] = '\0';
```

```
strcpy (middle, &pres[12]);
```

```
strncpy (first, &pres[7], 4);
first[4] = '\0';
```

J	o	h	n	\0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

String Terminology

- string length
  - in a character array, the number of characters before the first null character

J	o	h	n	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

- empty string
  - a string of length zero
  - the first character of the string is the null character

Scanning a Full Line

- For interactive input of one complete line of data, use the fgets function from stdio.

- Arguments: destination string, max characters to read, input

- Output: destination string or NULL if nothing read

- The \n character is stored if space.

fgets(<dest_string>, <num_chars>, <input>)

Scanning a Full Line with gets

```
char line[80];
printf("Type in a line of data.\n> ");
gets(line);
```

```
Type in a line of data.
> Here is a short sentence.
```

H	e	r	e		i	s		a		s	h	o	r	t		s	e	n	t	e	n	c	e	.	\0	. . .

subject to buffer overflow – we use fgets

String Comparison

a | b | l | u | e | \0 |

b | b | l | a | c | k | \0 |

c | b | l | u | e | \0 | ? | ? | ? | ? | ? |

d
| b | l | u | e | \0 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |

String Comparison

TABLE 8.2 Possible Results of strcmp(str1, str2)

Relationship	Value Returned	Example
str1 is less than str2	negative integer	str1 is "marigold" str2 is "tulip"
str1 equals str2	zero	str1 and str2 are both "end"
str1 is greater than str2	positive integer	str1 is "shrimp" str2 is "crab"

FIGURE 8.10 Sentinel-Controlled Loop for String Input

```
1.  printf("Enter list of words on as many lines as you like.\n");
2.  printf("Separate words by at least one blank.\n");
3.  printf("When done, enter %s to quit.\n", SENT);
4.
5.  for (scanf("%s", word);
6.      strcmp(word, SENT) != 0;
7.      scanf("%s", word)) {
8.      /* process word */
9.      . . .
10. }
```

Arrays of Pointers

- When sorting a list of strings, there is a lot of copying of characters from one memory cell to another.
  - 3 operations for every exchange
- C represents every array by its starting address.
- Consider an array of pointers, each element the address of a character string.

FIGURE 8.11   Exchanging String Elements of an Array

```
1. strcpy(temp, list[index_of_min]);
2. strcpy(list[index_of_min], list[fill]);
3. strcpy(list[fill], temp);
```

FIGURE 8.13

An Array
of Pointers
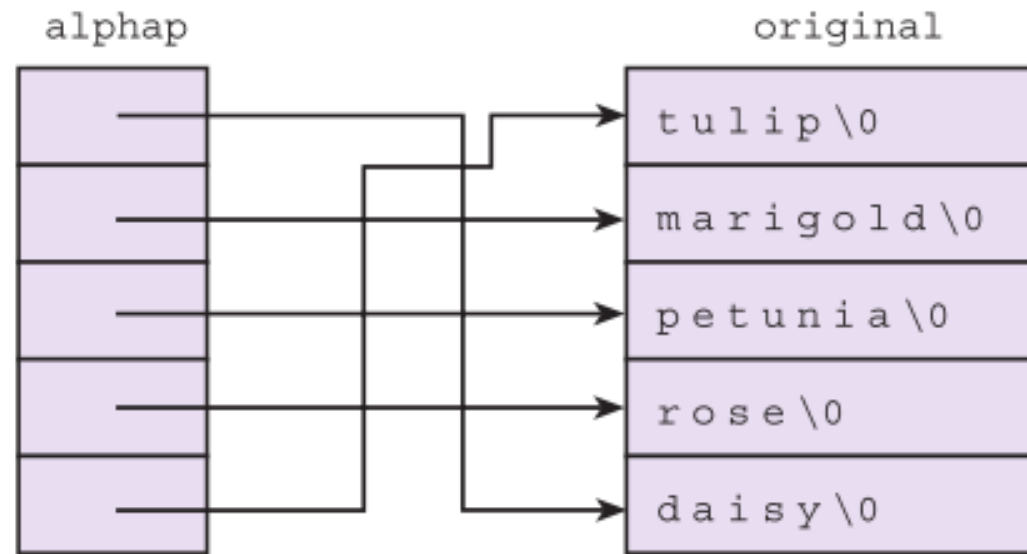
Concatenation

- strcat
  - appends source to the end of dest
  - assumes that sufficient space is allocated for the first argument to allow addition of the extra characters
    - s1 = "hello";
    - strcat(s1, "and more");

h	e	l	l	o	a	n	d		m	o	r	e	\0

Concatenation

- strncat
  - appends up to n characters of source to the end of dest, adding the null character if necessary
  - assumes that sufficient space is allocated for the first argument to allow addition of the extra characters
    - s1 = "hello";
    - strncat(s1, "and more", 5);

h	e	l	l	o	a	n	d		m	\0	?

String-to-Number and Number-to-String Conversions

TABLE 8.4   Review of Use of scanf

Declaration	Statement	Data (▌ means blank)	Value Stored
char t	scanf("%c", &t);	▌g \n A	 \n A
int n	scanf("%d", &n);	▌32▌ ▌▌−8.6 ▌+19▌	32 −8 19
double x	scanf("%lf", &x);	▌▌▌4.32▌ ▌−8▌ ▌1.76e−3▌	4.32 −8.0 .00176
char str[10]	scanf("%s", str);	▌▌hello\n overlengthy▌	hello\0 overlengthy\0 (overruns length of str)

String-to-Number and Number-to-String Conversions

TABLE 8.5 Placeholders Used with printf

Value	Placeholder	Output (▮ means blank)
'a'	%c	a
	%3c	▮▮a
	%-3c	a▮▮
-10	%d	-10
	%2d	-10
	%4d	▮-10
	%-5d	-10▮▮
49.76	%.3f	49.760
	%.1f	49.8
	%10.2f	▮▮▮▮▮49.76
	%10.3e	▮4.976e+01
"fantastic"	%s	fantastic
	%6s	fantastic
	%12s	▮▮▮fantastic
	%-12s	fantastic▮▮▮
	%3.3s	fan

String-to-Number and Number-to-String Conversions

- ## number to string: sprintf

  char s[10];
  int mon = 8, day = 23, year = 1914;
  sprintf(s, "%d/%d/%d", mon, day, year);

- ## string to number: sscanf

  int num;
  double val;
  char word[10];
  sscanf("85 96.2 hello", "%d%lf%s", &num, &val, word);

Things to remember

- Strings are just arrays of characters

- The string.h library provides functions for working with strings

- String variables are character pointers