CS 210 **October 20, 2016**
Day 14

Strings
Chapter 8

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
- use a – (minus) sign to force left justification
  - printf("%-20s\n", president);

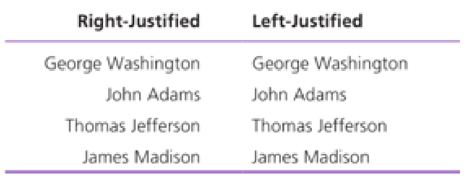


FIGURE 8.1 Right and Left Justification of Strings

Right-Justified	Left-Justified
George Washington	George Washington
John Adams	John Adams
Thomas Jefferson	Thomas Jefferson
James Madison	James Madison

```c
#include<stdio.h>
#pragma warning(disable:4996)
int main()
{
    char s1[] = "Hello Mom!";
    char s2[80];
    printf("%s\n", s1);
    //
    printf("Enter a string...");
    scanf_s("%s", s2, sizeof(s2)); //Requires buffer size
    printf("%s\n", s2);
    //
    printf("Enter a string...");
    scanf("%s", s2);   //Use pragma to enable
    printf("%s\n", s2);
}
```

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

String Assignment

- stcpy
  - copies the string that is its second argument into its first argument
    - strcpy(s1, “hello”);
  - subject to buffer overflow
- strncpy
  - take an argument specifying the number of characters to copy
  - if the string to be copies is shorter, the remaing characters stored are null
    - strncpy(s2, “inevitable”, 5);

```c
#include<stdio.h>
#include<string.h>
#pragma warning(disable:4996)

int main()
{
    char s1[] = "Hello Mom!";
    char s2[80];
    char s3[80];
    strcpy_s(s2, sizeof(s1), s1);
    printf("%s\n", s2);
    //
    strcpy(s2, s1);    //Use pragma for this
    printf("%s\n", s2);
    //
    strncpy(s3, s1, 3);
    s3[3] = '\0';
    printf("%s\n", s3);
    //
    strncpy(s3, &s1[2], 6);
    s3[6] = '\0';
    printf("%s\n", s3);
    //
    //             01234567890123456789012
    char name[] = "Matilda M. McGillicuddy";
    char first[20], middle[3], last[20];
    int n = sizeof(name);
    strncpy(last, &name[11], 12);
    strncpy(middle, &name[8], 2);
    strncpy(first, name, 7);
    last[12] = '\0';
    printf("%s, ", last);
    first[7] = '\0';
    printf("%s, ", first);
    middle[2] = '\0';
    printf("%s\n", middle);

}
```

Substrings

- a fragment of a longer string

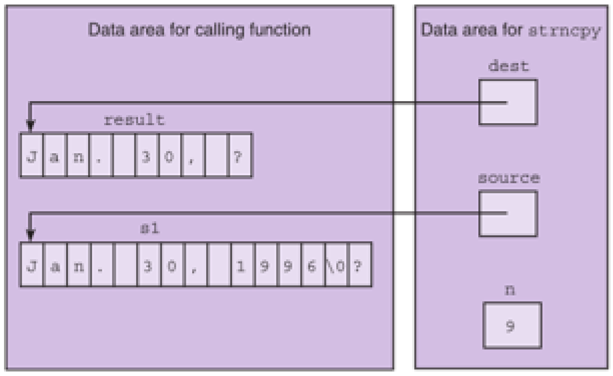


FIGURE 8.5 Execution of `strncpy(result, s1, 9);`

Substrings

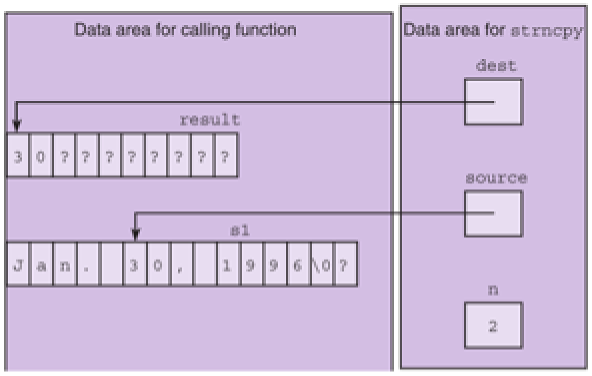
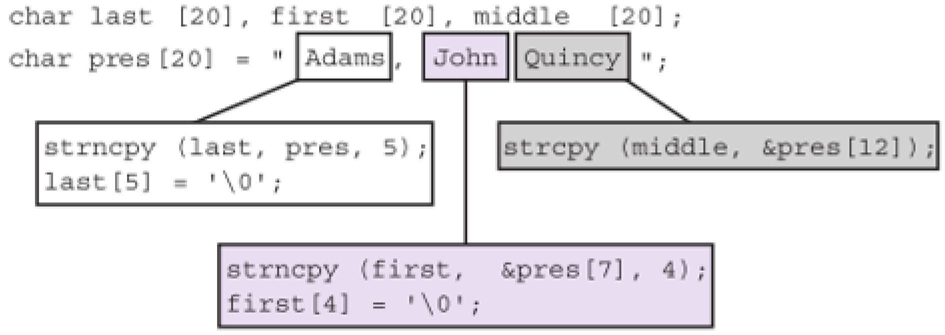


FIGURE 8.6 Execution of `strncpy(result, &s1[5], 2);`

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';
```

String Terminology

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

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	?

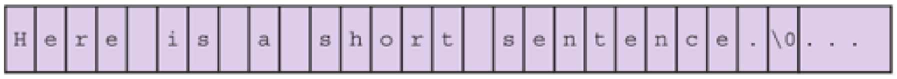
Scanning a Full Line

- For interactive input of one complete line of data, use the gets function.
- The \n character representing the <return> or <enter> key pressed at the end of the line is not stored.

Scanning a Full Line

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

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



```
#include<stdio.h>
#include<string.h>
int main()
{
    char line[80];
    int i, cnt = 0;
    printf("Enter a line of data... \n");
    gets(line);
    printf("%s\n", line);
    for(i=0;i<sizeof(line);i++)
       if(line[i] == ' ')
          cnt++;
    printf("There are %d spaces in the line.\n", cnt);
}
```

CS 210 **October 20, 2016**
Strings

Write a program which inputs a single line from the user as a string. Calculate and print the number of lower case letters a to z and print this number to the console.

Turn in a printed copy of your source file.

A sample output might look like this:

```
Enter a line of data...
aaabbbccczzz
a = 3
b = 3
c = 3
d = 0
e = 0
f = 0
g = 0
h = 0
i = 0
j = 0
k = 0
l = 0
m = 0
n = 0
o = 0
p = 0
q = 0
r = 0
s = 0
t = 0
u = 0
v = 0
w = 0
x = 0
y = 0
z = 3
Press any key to continue . . .
```