STRINGS II

Algorithms and Programming II

1

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*

2

Character Input/Output

- getchar
  - get the next character from the standard input source (that scanf uses)
  - does not expect the calling module to pass the address of a variable to store the input character
  - takes no arguments, returns the character as its result

```
ch = getchar()
```

3

Character Input/Output

- putchar
  - single-character output
  - first argument is a type int character code
  - recall that type char can always be converted to type in with no loss of information

```
putchar(‘a’);
```

4

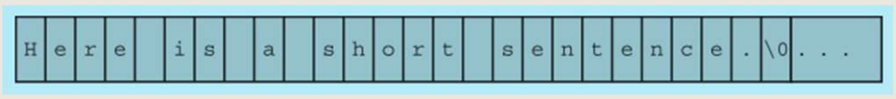
Scanning a Full Line

- 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.
```



String-Manipulation Functions of the String-Handling Library

- The string-handling library (<string.h>) provides many useful functions for manipulating string data (copying strings and concatenating strings), comparing strings, searching strings for characters and other strings, tokenizing strings (separating strings into logical pieces) and determining the length of strings.
- This section presents the string-manipulation functions of the string-handling library.
- Every function—except for strncpy—appends the null character to its result.

7

Function prototype	Function description
`char *strcpy(char *s1, const char *s2)`	*Copies* string `s2` into array `s1`. The value of `s1` is returned.
`char *strncpy(char *s1, const char *s2, size_t n)`	*Copies at most* `n` *characters* of string `s2` into array `s1` and returns `s1`.
`char *strcat(char *s1, const char *s2)`	*Appends* string `s2` to array `s1`. The first character of `s2` *overwrites the terminating null character* of `s1`. The value of `s1` is returned.
`char *strncat(char *s1, const char *s2, size_t n)`	*Appends at most* `n` *characters* of string `s2` to array `s1`. The first character of `s2` *overwrites the terminating null character* of `s1`. The value of `s1` is returned.

8

String-Manipulation Functions of the String-Handling Library

- Functions strncpy and strncat specify a parameter of type size_t, which is a type defined by the C standard as the integral type of the value returned by operator sizeof.
- Function strcpy copies its second argument (a string) into its first argument—a character array that you must ensure is large enough to store the string and its terminating null character, which is also copied.

9

String-Manipulation Functions of the String-Handling Library

- Function strncpy is equivalent to strcpy, except that strncpy specifies the number of characters to be copied from the string into the array.
- Function strncpy does not necessarily copy the terminating null character of its second argument.
- This occurs only if the number of characters to be copied is at least one more than the length of the string.

10

String-Manipulation Functions of the String-Handling Library

- Next figure uses strcpy to copy the entire string in array x into array y and uses strncpy to copy the first 14 characters of array x into array z.
- A null character ('\0') is appended to array z, because the call to strncpy in the program does not write a terminating null character (the third argument is less than the string length of the second argument).

11

```c
// Fig. 8.15: fig08_15.c
// Using functions strcpy and strncpy
#include <stdio.h>
#include <string.h>
#define SIZE1 25
#define SIZE2 15

int main(void)
{
   char x[] = "Happy Birthday to You"; // initialize char array x
   char y[SIZE1]; // create char array y
   char z[SIZE2]; // create char array z

   // copy contents of x into y
   printf("%s%s\n%s%s\n",
      "The string in array x is: ", x,
      "The string in array y is: ", strcpy(y, x));

```

12

```
    // copy first 14 characters of x into z. Does not copy null
    // character
    strncpy(z, x, SIZE2 - 1);

    z[SIZE2 - 1] = '\0'; // terminate string in z
    printf("The string in array z is: %s\n", z);
}
```

```
The string in array x is: Happy Birthday to You
The string in array y is: Happy Birthday to You
The string in array z is: Happy Birthday
```

13

Functions strcat and strncat

- Function strcat appends its second argument (a string) to its first argument (a character array containing a string).
- The first character of the second argument replaces the null ('\0') that terminates the string in the first argument.
- You must ensure that the array used to store the first string is large enough to store the first string, the second string and the terminating null character copied from the second string.
- Function strncat appends a specified number of characters from the second string to the first string.
- A terminating null character is appended to the result.
- Figure 8.16 demonstrates functions strcat and strncat.

14

```c
// Fig. 8.16: fig08_16.c
// Using functions strcat and strncat
#include <stdio.h>
#include <string.h>

int main(void)
{
   char s1[20] = "Happy "; // initialize char array s1
   char s2[] = "New Year "; // initialize char array s2
   char s3[40] = ""; // initialize char array s3 to empty

   printf("s1 = %s\ns2 = %s\n", s1, s2);

   // concatenate s2 to s1
   printf("strcat(s1, s2) = %s\n", strcat(s1, s2)  );

   // concatenate first 6 characters of s1 to s3. Place '\0'
   // after last character
   printf("strncat(s3, s1, 6) = %s\n", strncat(s3, s1, 6));

   // concatenate s1 to s3
   printf("strcat(s3, s1) = %s\n", strcat(s3, s1)  );
}
```

15

```
s1 = Happy
s2 = New Year
strcat(s1, s2) = Happy New Year
strncat(s3, s1, 6) = Happy
strcat(s3, s1) = Happy Happy New Year
```

16

Comparison Functions of the String-Handling Library

- This section presents the string-handling library's string-comparison functions, strcmp and strncmp.

Function prototype	Function description
`int strcmp(const char *s1, const char *s2);`	*Compares* the string `s1` with the string `s2`. The function returns `0`, less than `0` or greater than `0` if `s1` is equal to, less than or greater than `s2`, respectively.
`int strncmp(const char *s1, const char *s2, size_t n);`	*Compares up to n characters* of the string `s1` with the string `s2`. The function returns `0`, less than `0` or greater than `0` if `s1` is equal to, less than or greater than `s2`, respectively.

17

Comparison Functions of the String-Handling Library

- Figure 8.18 compares three strings using strcmp and strncmp.
- Function strcmp compares its first string argument with its second string argument, character by character.
- The function returns 0 if the strings are equal, a negative value if the first string is less than the second string and a positive value if the first string is greater than the second string.
- Function strncmp is equivalent to strcmp, except that strncmp compares up to a specified number of characters.
- Function strncmp does not compare characters following a null character in a string.
- The program prints the integer value returned by each function call.

18

```c
// Fig. 8.18: fig08_18.c
// Using functions strcmp and strncmp
#include <stdio.h>
#include <string.h>

int main(void)
{
   const char *s1 = "Happy New Year"; // initialize char pointer
   const char *s2 = "Happy New Year"; // initialize char pointer
   const char *s3 = "Happy Holidays"; // initialize char pointer

   printf("%s%s\n%s%s\n%s%s\n\n%s%2d\n%s%2d\n%s%2d\n\n",
      "s1 = ", s1, "s2 = ", s2, "s3 = ", s3,
      "strcmp(s1, s2) = ", strcmp(s1, s2) ,
      "strcmp(s1, s3) = ", strcmp(s1, s3) ,
      "strcmp(s3, s1) = ", strcmp(s3, s1) );

   printf("%s%2d\n%s%2d\n%s%2d\n",
      "strncmp(s1, s3, 6) = ", strncmp(s1, s3, 6) ,
      "strncmp(s1, s3, 7) = ", strncmp(s1, s3, 7) ,
      "strncmp(s3, s1, 7) = ", strncmp(s3, s1, 7) );
}
```

19

```
s1 = Happy New Year
s2 = Happy New Year
s3 = Happy Holidays

strcmp(s1, s2) =  0
strcmp(s1, s3) =  1
strcmp(s3, s1) = -1

strncmp(s1, s3, 6) =  0
strncmp(s1, s3, 7) =  1
strncmp(s3, s1, 7) = -1
```

20

Comparison Functions of the String-Handling Library

- To understand just what it means for one string to be “greater than” or “less than” another, consider the process of alphabetizing a series of last names.
- The reader would, no doubt, place “Jones” before “Smith,” because the first letter of “Jones” comes before the first letter of “Smith” in the alphabet.

21

Comparison Functions of the String-Handling Library

- But the alphabet is more than just a list of 26 letters—it’s an ordered list of characters.
- Each letter occurs in a specific position within the list.
- “Z” is more than merely a letter of the alphabet; “Z” is specifically the 26th letter of the alphabet.
- How do the string comparison functions know that one particular letter comes before another?
- All characters are represented inside the computer as numeric codes in character sets such as ASCII and Unicode; when the computer compares two strings, it actually compares the numeric codes of the characters in the strings.

22