System Design and Programming II

CSCI – 1943

David L. Sylvester, Sr., Professor

Chapter 10

Characters, Strings,
and the **string** Class

Character Testing

The C++ library provides several functions for testing characters. To use these functions you must include the cctype header file.

These libraries provide several functions that allow you to test the value of a character. These functions test a single char argument and return either true or false. They actually return an int value (0 – false, nonzero – true).

Ex:
```
char letter = 'a';
if (isupper(letter));
    cout << "Letter is uppercase.\n";
else
    cout << "Letter is lowercase.\n";
```

will output **Letter is lowercase.**

Character Testing Functions	
Character Function	Description
isalpha	Returns true (nonzero number) if the argument is a letter of the alphabet. Return 0 if the argument is not a letter
isalnum	Returns true (nonzero number) if the argument is a letter of the alphabet or a number. Otherwise, it returns 0.
isdigit	Returns true (nonzero number) if the argument is a digit from 0 through 9. Otherwise, it returns 0.
islower	Returns true (nonzero number) if the argument is a lowercase letter. Otherwise, it returns 0.
isprint	Returns true (nonzero number) if the argument is a printable character (including a space). Otherwise, it returns 0.
ispunct	Returns true (nonzero number) if the argument is a printable character other than a digit, letter or space. Otherwise, it returns 0.
isupper	Returns true (nonzero number) if the argument is an uppercase letter. Otherwise, it returns 0.
isspace	Returns true (nonzero number) if the argument is a whitespace character. (i.e. space ‘ ‘, vertical tab ‘\v’, newline ‘\n’, tab ‘\t’). Otherwise, it returns 0.

Character Functions – Sample Program

```
#include <iostream>
// This program demonstrates some
// character-testing functions.
#include <iostream>
#include <cctype>
using namespace std;

int main()
{
char input;
cout << “Enter any character: ";
cin.get(input);
cout << "The character you entered is: "
<< input << endl;
if (isalpha(input))
cout << "That's an alphabetic character.\n";
if (isdigit(input))
cout << "That's a numeric digit.\n";
if (islower(input))
cout << "The letter you entered is
lowercase.\n";
if (isupper(input))
cout << "The letter you entered is
uppercase.\n";
if (isspace(input))
cout << "That's a whitespace character.\n";
return 0;
}
```

Character Functions – Sample Program

```
// This program test a customer number
// to determine whether it is in the
// proper format.
#include <iostream>
#include <cctype>
using namespace std;

// Function prototype
bool testNum(char[], int);

int main()
{
    const int SIZE = 8;// Array size
    char customer[SIZE];// To hold a customer number

    // Get the customer number.
    cout << "Enter a customer number in the form ";
    cout << "LLLNNNN\n";
    cout << "(LLL = letters and NNNN = numbers): ";
    cin.getline(customer, SIZE);

    // Determine whether it is valid.
    if (testNum(customer, SIZE))
        cout << "That's a valid customer number.\n";
    else
    {
        cout << "That is not the proper format for the ";
        cout << "customer number.\nHere is an example\n";
        cout << "  ABC1234\n";
    }
    return 0;
}

// Definition of Function testNum.

bool testNum(char custNum[], int size)
{
    int count;// Loop counter

    // Test the first three characters for alphabetic letters.
    for (count = 0; count < 3; count++)
    {
        if (!isalpha(custNum[count]))
            return false;
    }

    // Test the remaining characters for numeric digits.
    for (count = 3; count < size -1; count++)
    {
        if (!isdigit(custNum[count]))
            return false;
    }
    return true;
}
```

Character Case Conversion

The C++ library provides two functions, toupper and tolower, for converting the case of a character. These functions are also prototyped in the header file cctype, so be sure to include it.

Additional Character Functions	
toupper	Returns the uppercase equivalent of its argument.
tolower	Returns the lowercase equivalent of its argument.

Note: These functions accept a single character argument.

Ex: cout << toupper ('a'); outputs A

If the argument is already an uppercase letter, toupper return is unchanged. Any nonletter argument passed to toupper is returned as it is.

```
Ex:     cout << toupper('*')        // Displays *
        cout << toupper('&')        // Displays &
        cout << toupper('%')        // Displays %
```

toupper and tolower do not actually cause the character argument to change (change the stored value), they simply return the upper or lower equivalent of the argument.

```
Ex:     char letter = 'A';
        cout << tolower(letter) << endl;
        cout << letter;
```

Will output a

Will output A

Character Functions – Sample Program

```
// This program calculates the area of a circle.
// It asks the user if he or she wishes to continue.
// A loop that demonstrates the toupper function repeats
// until the user enters 'y', 'Y' or 'n', 'N'.
#include <iostream>
#include <cctype>
#include <iomanip>
using namespace std;

char goAgain;// To hold Y or N

int main()
{
    const double PI = 3.14159;// Constant for pi
    double radius;// The circle's radius

    cout << "This program calculates the area of a
            circle.\n";
    cout << fixed << setprecision(2);
     do
    {
        // Get the radius and display the area.
        cout << "Enter the circle's radius: ";
        cin >>radius;
        cout << "The area is "<< (PI * radius * radius);
        cout << endl;

        // Does the user want to do this again?
        cout << "Calculate another?  (Y or N) ";
        cin >> goAgain;

        // Validate the input.
        while (toupper(goAgain) != 'Y' && toupper(goAgain)
                != 'N')
        {
                cout << "Please enter Y or N: ";
                cin >> goAgain;
        }

    } while (toupper(goAgain) == 'Y');
    return 0;
}
```

Prompts the user for the radius

Calculates and output area of circle

Makes only 2 comparison rather than 4 (Y,y,N,n)

Review of the Internal Storage of C-Strings

String in a generic term that describes any consecutive sequence of characters.

Ex: word
sentence "Have a nice day."
person's name
title of a song

A string may be constant or variable in nature, and may be stored in a variety of ways.

C-strings describes a string whose characters are stored in consecutive memory locations and are followed by a null character, or null terminator. The null terminator marks the end of the C-string. Without it a function has no way of knowing the length of a C-string argument.

T	O	M	O	R	R	O	W	\0

String Literals

A string literal or string constant is enclosed in a set of double quotes.

Ex: “Have a nice day.”

“What is your name?”

“John Smith”

“Please enter your age:”

“Part Number 45q1798”

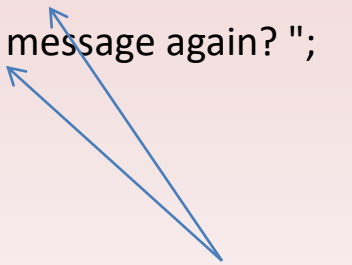
All of a program’s string literals are stored in memory as C-strings, with the null terminator automatically appended.

Character Functions – Sample Program

```
// This program contains string literals.
#include <iostream>
using namespace std;

int main()
{
    char again;

    do
    {
        cout << "C++ programming is great fun!" << endl;
        cout << "Do you want to see the message again? ";
        cin >> again;
    } while (again == 'Y' || again == 'y');
    return 0;
}
```



Although the strings are not stored in arrays, they are still part of the program's data. The first string occupies 30 bytes of memory and the second occupies 39 bytes. Both sizes include the null terminator.

The string literals on the previous page would be represented as such:

C	+	+		p	r	o	g	r	a	m	m	i	n	g		i	s		g	r	e	a	t	
f	u	n	!	\0																				

D	o		y	o	u		w	a	n	t		t	o		s	e	e		t	h	e		m	e
s	s	a	g	e		a	g	a	i	n	?		\0											

It is important to realize that a string literal has its own storage location, just like a variable or an array. When a string literal appears in a statement, it's actually its memory address that C++ uses.

Ex: cout << "Do you want to see the message again? ";

In this statement, the memory address of the string literal "Do you want to see the message again? " is passed to the cout object. cout displays the consecutive characters found in this address and stop displaying characters when a null terminator is encountered.

Strings Stored in Array

When defining a character array for holding a C-string, be sure the array is large enough for the null terminator.

Ex: `char company[12]` // holds no more than 11 characters

String input can be performed by using the cin object. It allows a string to be entered that has no whitespace characters into the company array. If a user enters more than 11 characters, cin will write past the end of the array, because it has no way of knowing that the variable company has 12 elements.

Ex: `cin >> company` ← Because company is an array and is being used without the brackets or subscripts, it indicate the address in memory where string is to be stored.

The cin problem can be solved by using the cin's getline member function.

Ex:
```
char line[80];
cin.getline(line,80);
```

Gets a line of input (including whitespace characters) and store it in the line array.

Starting address of line array

Number of character to accept including null terminator.

cin will read 79 characters, or until the user presses the **[ENTER]** key, whichever comes first. cin will automatically append the null terminator.

Sample Program

```
// This program displays a string stored
// in a char array.
#include <iostream>
using namespace std;

int main()
{
    const int SIZE = 80;// Array size
    char line[SIZE];// To hold a line of input
    int count = 0;// Loop counter variable

    // Get a line of input.
    cout << "Enter a sentence of no more than "
         << (SIZE - 1) << " characters.\n";
    cin.getline(line,80);

    // display the input one character at a time.
    cout << "The sentence you entered is:\n";
    while(line[count] != '\0')
        {
            cout << line[count];
            count ++;
         }
         return 0;
}
```

Library Functions for Working with C-strings

The C++ library has numerous functions for handling C-strings. They perform various tests and manipulations, and require that the ***cstring* header file** be included.

	Function	Description (*Accepts 1 C-string or pointer; **Accepts 2 C-strings or pointers to a C-string)
*	strlen	Returns the length of the C-string (not including the null terminator) Ex: **len = strlen(name);**
**	strcat	Appends the contents of the second string to the first C-string. (The first string is altered, the second string is unchanged.) Ex: **strcat(string1, string2);**
**	strcpy	Copies the second C-string to the first C-string. The second is unchanged. Ex: **strcpy(string1, string2);**
**	strncat	The third argument, an integer, indicates the maximum number of characters to copy from the second C-string to the first C-string. Ex: **strncat(string1, string2, n);**
**	strncpy	The third argument, an integer, indicates the maximum number of characters to copy from the second C-string to the first C-string. (string1 padded with \0 if n > string 2; Ex: **strncpy(string1, string2, n);**
**	strcmp	If string1 and string2 are the same, returns 0. If string2 alphabetically greater will return -1, if string 2 alphabetically less than it will return 1. Ex: **if (strcmp(string1, string2))**
**	strstr	Searches for the first occurrence of string2 in string1. If found returns a pointer to it. Otherwise returns a NULL pointer (address 0) Ex: **cout << strstr(string1, string2);**

The strlen Function

Ex: char name[50] = “Thomas Edison”;

int length;

length = strlen(name);

The strlen function accepts a pointer to a C-string as its argument. It returns the length of the string, which is the number of characters up to, but not including the null terminator.

The variable length will have the value integer value 13.

Note: Do not confuse the length of the string with the size of the array. The only information being passed to strlen is the beginning address of a C-string. It does not know where the array ends, so it looks for the null terminator to indicate the end of the string.

Using strlen with a literal string.

Ex: length = strlen(“Thomas Edison”);

The strcat Function

The strcat function concatenates, or appends one string to another.

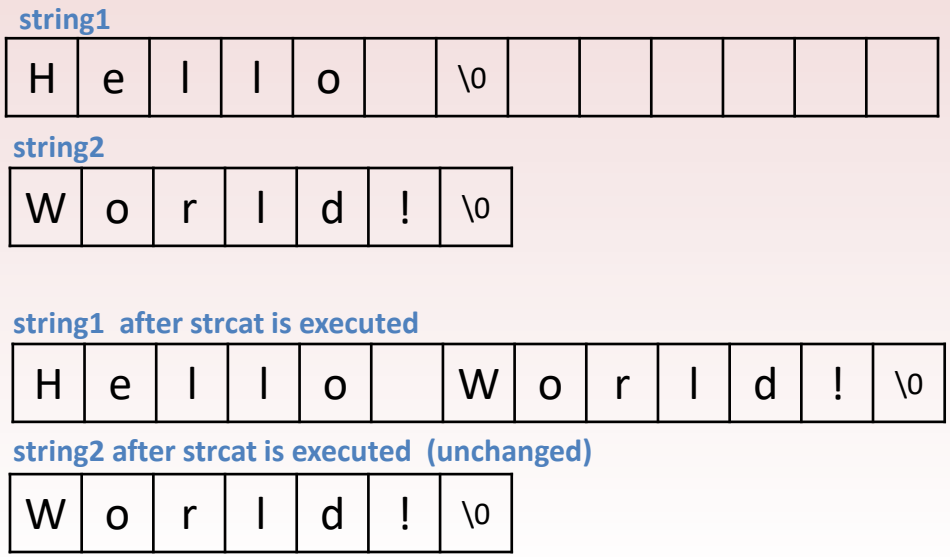
Ex:

```
char string1[13] = "Hello ";
char string2[7] = "World!";
cout << string1 << endl;
cout << string2 << endl;
strcat(string1, string2);
cout << string1;
```

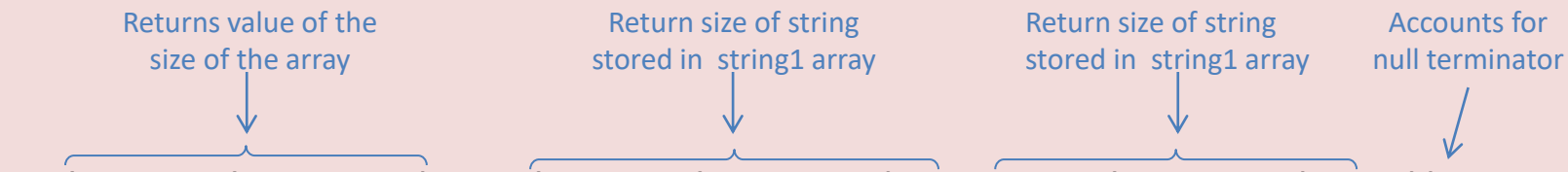
Appends string1 with string2

Code produces this output

```
Hello
World!
Hello World!
```



Sample program segment that uses sizeof operator to test an array's size before strcat is called.




```
if (sizeof(string1) >= (strlen(string1) + strlen(string2) + 1))
    strcat(string1, string2);
else
    cout << "String1 is not large enough for both strings.\n";
```

The strcpy Function

Unlike copying arrays where you have to copy one element at a time, usually using a for loop, with strings the strcpy function can be used to copy one string into another.

```
Ex:     char name[20];
        strcpy(name, "Albert Einstein");
```

The second argument of the strcpy function is copied into the first argument including the null terminator. Hence, copying "Albert Einstein" into the name array.

If anything is already stored in the location referenced by the first argument, it is overwritten.

```
Ex:     char string1[10] = "Hello", string2[10] = "World";
        cout << string1 << endl;
        cout << string2 << endl;
        strcpy(string1, string2);
        cout << string1 << endl;
        cout << string2 << endl;
```

Overwrites string1 with string2

Code produces this output

```
Hello
World!
World!
World!
```

The strncat and strncpy Function

Because the strcat and strcpy functions can potentially overwrite the bounds of an array, it is possible to write unsafe code. As an alternative, you should use the strncat and strncpy wherever possible.

The strncat function works like strcat, except it takes a third argument specifying the maximum number of characters from the second string to append to the first.

Ex: strncat(string1, string2, 10);

Copies no more than 10 characters from string2 into string1.

Sample program shows an example of calculating the maximum number of characters that can be appended to an array.

Ex:

```
int maxChars;
char string1[17] = "Welcome ";
char string2[18] = "to North Carolina";
maxChars = sizeof(string1) – (strlen(string1) + 1);
strncat(string1, string2, maxChars);
cout << string1 << endl;
```

Accounts for null terminator

maxChars = 17 – (8 + 1)
maxChars = 8

Calling strncpy is similar to calling strcpy, except you pass a third argument specifying he maximum number of characters from the second string to copy to the first.

Ex: strncpy(string1, string2, 5);

When executed, strncpy will copy no more than five characters from string2 to string1.

- If the number of characters is less than or equal to the length of string2, a null terminator is not appended to string1.
- If the specified number of characters is greater than the length of string2, then string 1 is padded with null terminators (up to the specified number of characters)

strncpy example:

```
int maxChars;
char string1[11];
char string2[ ] = “I love C++ programming!”;
maxChars = sizeof(string1 -1);
strncpy(string1, string2, maxChars);
// Put the null terminator at the end.
string1[10] = ‘\0’;
cout << string1 << endl;
```

The above statement adds the null terminator at the end of string1. This is done because maxChars was less than the length of string2, and strncpy did not automatically place a null terminator there.

The strstr Function

The strstr function searches for a string inside of a string. The function's first argument is the string to be searched, and the second argument is the string to look for. If the function finds the second string inside the first, it returns the address of the occurrence of the second string within the first string. Otherwise it returns the address 0, or the NULL address.

Ex

```
char arr[ ] = "Four score and seven years ago";
char *strPtr;
cout << arr << endl;
strPtr = strstr(arr, "seven");    // search for "seven"
cout << strPtr << endl;
```

Will output: Four score and seven years ago

Will output: seven years ago

In this code, strstr locates the string "seven" inside the string "Four score and seven years ago." It returns the address of the first character in "seven" which will be stored in the pointer variable strPtr.

Sample Program

```
// This program uses the strstr
// function to search an array.
#include <iostream>
#include <cstring>// For strstr
using namespace std;

int main()
{
    // Constants for array lengths
    const int NUM_PRODS = 5;      // Number of   products
    const int LENGTH = 27;        // String length

    // Array of products
    char products[NUM_PRODS][LENGTH] =
              {"TV327 31-inch Television",
              "CD257 CD Player",
              "TA677 Answering Machine",
              "CS109 Car Stereo",
              "PC955 Personal Computer"};

    char lookUp[LENGTH];          // To hold user's input
    char *strPtr = NULL;          // To point to the found product
    int index;                    // Loop counter

    // Prompt the user for  a product number.
    cout << "\tProduct Database\n\n";
    cout << "Enter a product number to
            search for: ";
    cin.getline(lookUp,LENGTH);

    // Search the array for a match substring
    for (index = 0; index < NUM_PRODS; index++)
    {
        strPtr = strstr(products[index], lookUp);
        if (strPtr != NULL)
            break;
    }

    // If a matching substring was found,
    // display the product info.
    if (strPtr != NULL)
        cout << products[index] << endl;
    else
        cout << "No matching product was
                found.\n";
    return 0;
}
```

String/Numeric Conversion Function

The C++ library provides functions for converting a string representation of a number to a numeric data type and vice versa. These functions require the **cstdlib header file** to be included. The string “26792” isn’t actually a number, but a series of ASCII codes representing the individual digits of the number. It uses six bytes of information including the null terminator and it is not possible to perform mathematical operations with it.

String Conversion Functions	
Function	Description
atoi	Converts the C-string to an integer and return that value. Ex: num = atoi(“4569”);
atol	Converts the C-string to a long integer and return that value. Ex: lnum = atol(“500000”);
atof	Converts the C-string to a double and returns that value. fnum = atof(“3.14159”);
itoa	Converts integer to C-string. (8 = octal, 10 = decimal , 16 = hexadecimal) itoa(value, string, base);

The atoi Function

Ex:
```
int num;
num = atoi(“1000”);
```

The atol Function

Ex:
```
long bignum;
bignum = atol(“500000”);
```

The atof Function

Ex:
```
double;
num = atof(“12.67”);
```

The itoa Function

Ex:
```
char numArray[10]
itoa(1200, numArray, 10);
cout << numArray << endl;
```

The C++ string Class

Standard C++ provides a special data type for storing and working with strings.

The string class is an abstract data type. This means it is not a built-in, primitive data type line int or char. Instead, it is a programmer-defined data type that accompanies the C++ language. It provides many capabilities that make storing and working with strings easy and intuitive.

The first step in using the string class is to #include the string header file. This accomplished with the following preprocessor directive:

#include <string>

Now you are ready to define a string object. Defining a string object is similar to defining a variable of a primitive type.

Ex: **string movieTitle;**

The C++ string Class

You assign a string value to the movieTitle object with the assignment operator, as

movieTitle = “Wheels of Fury”;