COP 3223: C Programming Spring 2009

Strings In C – Part 4

Instructor: Dr. Mark Llewellyn

markl@cs.ucf.edu

HEC 236, 407-823-2790

http://www.cs.ucf.edu/courses/cop3223/spr2009/section1

School of Electrical Engineering and Computer Science University of Central Florida



The Character Handling Library

- Although we are primarily dealing with strings, processing of character data within the strings is an important and often utilized function of many application programs.
- The character handling functions are found in the <ctype.h> standard library.
- Some of the more common character handling functions are shown in the tables on the next two pages.
- Following the tables are several example programs that illustrate some of the character handling functions in <ctype.h>.



Some Of The Functions In <ctype.h>

Function Prototype	Function Description
int isdigit (int c);	Returns a true (non-zero) value if c is a digit and 0 (false) otherwise.
int isalpha (int c);	Returns a true value if c is a letter and 0 (false) otherwise.
int isalnum (int c);	Returns a true value if c is a digit or a letter and 0 (false) otherwise.
int islower (int c);	Returns a true value if c is a lowercase letter and 0 (false) otherwise.
int isupper (int c);	Returns a true value if c is an uppercase letter and 0 (false) otherwise.
int tolower (int c);	If c is an uppercase letter, this function returns c as a lowercase letter. Otherwise, the function returns the argument unchanged.
int toupper (int c);	If c is a lowercase letter, this function returns c as an uppercase letter. Otherwise, the function returns the argument unchanged.

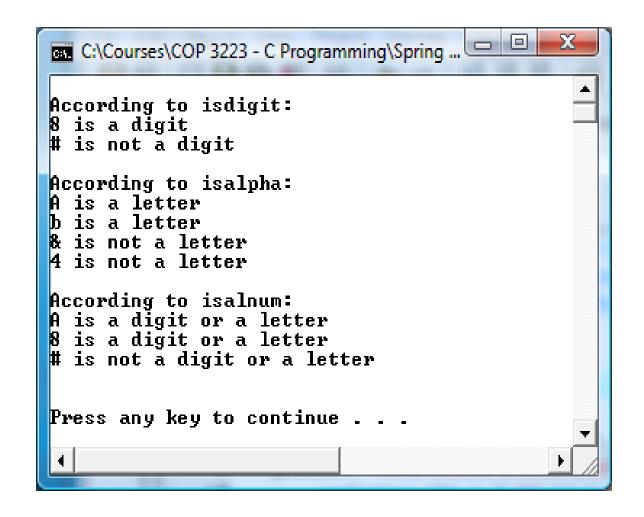


Some Of The Functions In <ctype.h>

Function Prototype	Function Description
int isspace (int c);	Returns a true (non-zero) value if c is a white-space character. This includes: newline ('\n'), space (' '), form feed ('\F'), carriage return ('\r'), horizontal tab ('\t'), or vertical tab ('\v'). Otherwise a value of 0 (false) is returned.
int iscntrl (int c);	Returns a true value if c is a control character and 0 (false) otherwise.
int ispunct (int c);	Returns a true value if c is a printing character other than a space, a digit, or a letter and 0 (false) otherwise. (This function is basically returning true for punctuation marks.)
int isprint (int c);	Returns a true value if c is a printing character including a space ('') and 0 (false) otherwise.
int isgraph (int c);	Returns a true value if c is a printing character other than a space ('') and 0 (false) otherwise.







An aside on the conditional operator?

The previous example program utilizes the conditional operator?. This C operator is a sometimes useful shorthand replacement for an if-else control statement.

The conditional operator is the only ternary operator (ternary means 3-way) in C. The conditional operator requires 3 operands. The operands together with the conditional operator form a conditional expression.

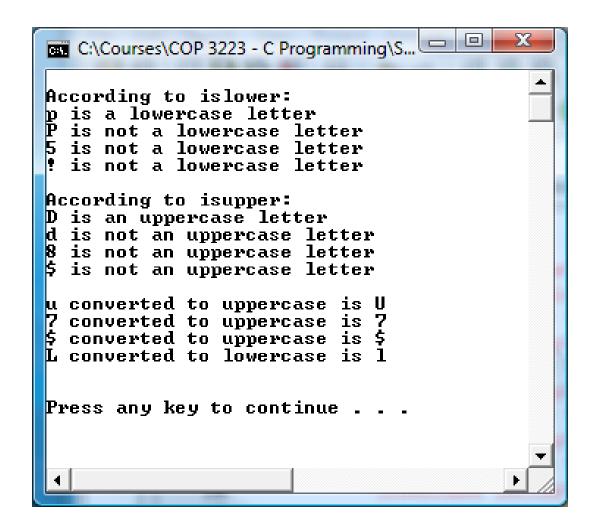
The first operand is a condition, the second operand is the value for the entire conditional expression if the condition is true and the third operand is the value for the entire conditional expression if the condition is false.

```
The general form is: condition ? operand : operand;
```

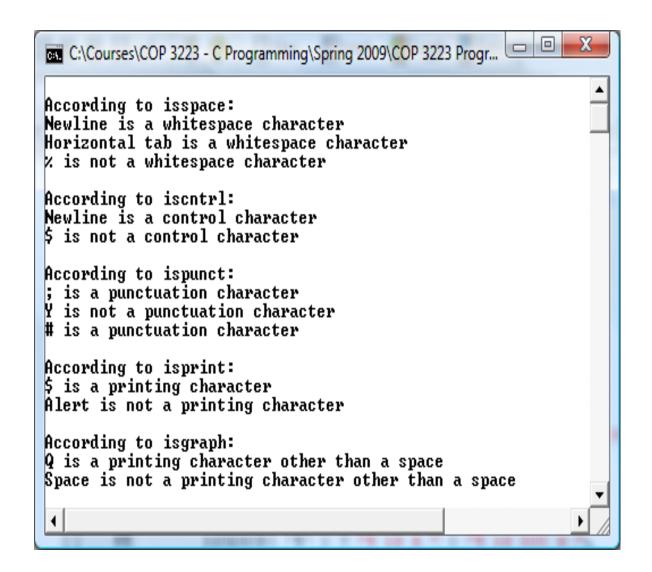
An example is: grade >= 60 ? printf("passed\n"): printf('failed\n");



```
7 int main()
 8 {
9
     printf( "\n%s\n%s%s\n%s%s\n%s%s\n%s%s\n\n",
10
              "According to islower:",
11
              islower( 'p' ) ? "p is a " : "p is not a ",
12
              "lowercase letter",
13
              islower( 'P' ) ? "P is a " : "P is not a ",
14
              "lowercase letter",
15
              islower( '5' ) ? "5 is a " : "5 is not a ",
16
              "lowercase letter",
17
              islower( '!' ) ? "! is a " : "! is not a ",
18
              "lowercase letter" );
19
20
     printf( "%s\n%s%s\n%s%s\n%s%s\n%s%s\n\n",
21
              "According to isupper: ",
22
              isupper( 'D' ) ? "D is an " : "D is not an ",
23
              "uppercase letter",
24
              isupper('d') ? "d is an ": "d is not an ",
25
              "uppercase letter",
26
              isupper( '8' ) ? "8 is an " : "8 is not an ",
27
              "uppercase letter",
28
              isupper( '$' ) ? "$ is an " : "$ is not an ",
29
              "uppercase letter" );
30
31
     printf( "%s%c\n%s%c\n%s%c\n\n\n",
              "u converted to uppercase is ", toupper( 'u' ),
32
33
              "7 converted to uppercase is ", toupper( '7' ),
34
              "$ converted to uppercase is ", toupper( '$' ),
35
              "L converted to lowercase is ", tolower( 'L' ) );
36
      system("PAUSE");
37
      return 0;
```



```
7 int main()
8 {
9
     printf( "\n%s\n%s%s%s\n%s%s\n\n",
10
          "According to isspace: ", "Newline", isspace( '\n' ) ? " is a " :
11
         " is not a ", "whitespace character", "Horizontal tab",
12
         isspace( '\t' ) ? " is a " : " is not a ", "whitespace character",
13
         isspace( '%' ) ? "% is a " : "% is not a ", "whitespace character" );
14
15
     printf( "%s\n%s%s%s\n%s%s\n\n", "According to iscntrl:",
16
          "Newline", iscntrl( '\n' ) ? " is a " : " is not a ",
         "control character", iscntrl( '$' ) ? "$ is a " :
17
18
         "$ is not a ", "control character" );
19
20
     printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
21
          "According to ispunct: ", ispunct( ';' ) ? "; is a " : "; is not a ",
22
         "punctuation character", ispunct( 'Y' ) ? "Y is a " : "Y is not a ",
23
         "punctuation character", ispunct( '#' ) ? "# is a " : "# is not a ",
24
         "punctuation character" );
25
26
     printf( "%s\n%s%s\n%s%s%s\n\n", "According to isprint:",
27
         isprint( '$' ) ? "$ is a " : "$ is not a ", "printing character",
28
         "Alert", isprint( '\a' ) ? " is a " : " is not a ",
29
         "printing character" );
30
31
     printf( "%s\n%s%s\n%s%s\n\n\n", "According to isgraph:",
32
         isgraph( 'Q' ) ? "Q is a " : "Q is not a ",
33
         "printing character other than a space",
34
         "Space", isgraph( ' ' ) ? " is a " : " is not a ",
35
         "printing character other than a space" );
36
37
     system("PAUSE");
```



Using String Conversion Functions

- Another sometimes useful operation that needs to be performed on strings is their conversion into numeric formats.
- For example, converting the string "1234" into the integer value 1234.
- Again, the C standard library has several functions that perform this task depending on the numeric type of the conversion.
- These functions are included in the <stdlib.h> standard library.
- The table on the next page lists some of the more common string conversion functions in this library.
- Care should be used when dealing with such functions, particularly in converting to integer types to ensure that overflow does not occur. If it is possible to do so, the application should normally read numeric types as numeric input rather than convert using these functions.



Some Of The String Conversion Functions In

<stdlib.h>

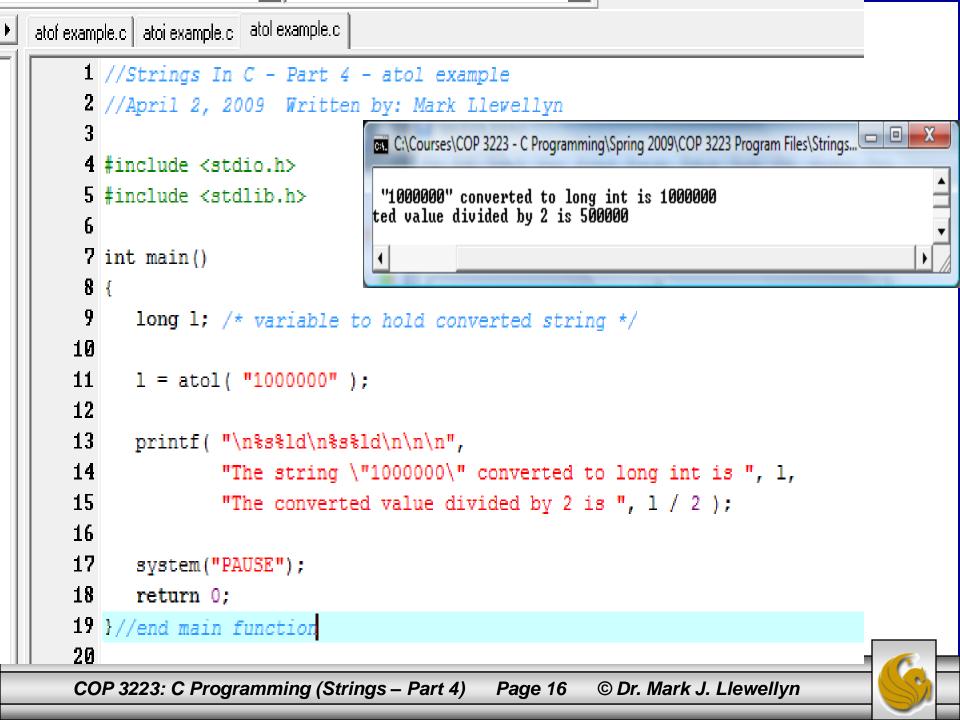
Function Prototype	Function Description
double atof (const char *nPtr);	Converts the string pointed to by nPtr to a double.
int atoi (const char *nPtr);	Converts the string pointed to by nPtr to an int.
long atol (const char *nPtr);	Converts the string pointed to by nPtr to a long int.

- Example program using each of these three functions are shown on the next page.
- Notice in each of the programs that after the string is converted to a numeric type, the converted numeric value is used in an expression to illustrate the conversion.



```
atof example.c
     1 //Strings In C - Part 4 - Using atof
                                                   C:\Courses\COP 3223 - C Programming\Spring 2009\COP 322...
     2 //April 2, 2009 Written by: Mark Lleve
     3
                                                    The string "99.0" converted to double is 99.000
     4 #include <stdio.h>
                                                    The converted value divided by 2 is 49.500
     5 #include <stdlib.h>
     6
                                                    Press any key to continue . . .
     7 int main()
    8 {
          double d; // variable to hold converted string
   10
   11
          d = atof("99.0");
   12
   13
          printf( "\n%s%.3f\n%s%.3f\n\n\n",
   14
                   "The string \"99.0\" converted to double is ", d,
   15
                   "The converted value divided by 2 is ",
   16
                   d / 2.0);
   17
          system("PAUSE");
   18
          return 0;
   19 }//end main function
   2И
```

```
atof example.c atoi example.c
                                           C:\Courses\COP 3223 - C Programming\Spring 2009\COP 3223 Progra...
     1 //Strings In C - Part 4 - atoi
     2 //April 2, 2009 Written by: Ma
                                           The string "2593" converted to int is 2593
                                           The converted value minus 593 is 2000
     4 #include <stdio.h>
     5 #include <stdlib.h>
                                           The string "123456789123" converted to int is -1097262461
                                           The converted value minus 500 is -1097262961
     6
     7 int main()
     8 {
     9
          int i; // variable to hold converted string
    10
    11
          i = atoi("2593");
    12
    13
          printf( "\n%s%d\n%s%d\n\n\n",
    14
                    "The string \"2593\" converted to int is ", i,
    15
                    "The converted value minus 593 is ", i - 593 );
    16
          //this one will cause an overflow
    17
          i = atoi( "123456789123" );
    18
    19
          printf( "\n%s%d\n%s%d\n\n\n",
    20
                    "The string \"123456789123\" converted to int is ", i,
    21
                    "The converted value minus 500 is ", i - 500 );
    22
          system("PAUSE");
    23
          return 0:
    24 }//end main function
      COP 3223: C Programming (Strings – Part 4)
                                                Page 15
                                                           © Dr. Mark J. Llewellyn
```



Arrays Of Strings

- As a final example dealing with strings, we'll look at a program that uses a two-dimensional array holding strings. In other words, an array of strings.
- In the version shown here, we'll treat the 2-dimensional array in much the same was that we have treated other arrays, using implicit pointers being passed to functions needing access to the array.
- We'll look in some more detail at pointer arithmetic later and we'll revisit this problem and use explicit pointer references.
- The following program reads lines of text, the number of which is unknown in advance, from a file and stores the text in a 2-dimensional array of strings.



Iyrics - Notepad

File Edit Format View Help

It's never been easy for me

The data file

To find words to go along with a melody But this time there's actually something, on my mind So please forgive these few brief awkward lines

Since I met you, my whole life has changed It's not just my furniture you?ve rearranged I was living in the past, but somehow you've brought me back And I haven't felt like this since before Frankie said relax

And while I know, based on my track record, I might not seem like the safest bet All I'm asking you is don't write me off, just yet

For vears I've been telling myself the same old story That I'm happy to live off my so-called former glories But you've given me a reason to take another chance Now I need you, despite the fact that you've killed all my plants

And though I know, I've already blown more chances Than anyone should ever get All I'm asking you is don?t write me off, just yet Don't write me off just yet



```
2 //This program reads in several lines of text from a file and stores the text
 3 //in a two-dimensional array.
 4 //NOTE: This version of the program treats the lyrics as a 2-dimensional array
 5 // of strings.
 6 //March 20, 2009 Written by: Mark Llewellyn
 7 #include <stdio.h>
 8 #include <string.h>
 9 #define MAX LENGTH 81
10 #define MAX LINES 40
                  MAX LINES
11
12 int readLyrics(char music[MAX LINES][MAX LENGTH])
13 {
14
      FILE *inFilePtr; //declare input file pointer
    char line[MAX LENGTH]; //a line of the lyrics
15
16
      int index = 0; //a line counter
17
      char *discard; //return pointer not used
18
19
      if ( (inFilePtr = fopen("lyrics.dat", "r") ) == NULL ) {
20
         printf("Sorry, the file could not be opened\n");
21
22
      else {
23
          fgets(line, MAX LENGTH, inFilePtr);
24
          while ( !feof(inFilePtr) ) {
25
               discard = strcpy(music[index], line);
26
              index++;
27
               fgets(line, MAX LENGTH, inFilePtr);
28
          }//end while stmt
29
           return index:
30
       }//end else
31 }//end function readLyrics
```

arrays of strings - music lyrics.c.

```
[*] arrays of strings - music lyrics.c.
   32
   33
   34
   35
   36
   37 void printLyrics(char music[MAX LINES][MAX LENGTH], int counter)
   38 {
   39
            char line[MAX LENGTH]; //a line of lyrics
   40
            int index; //loop control
   41
   42
            for (index = 0; index < counter; index++) {</pre>
   43
                puts(music[index]);
   44
            }//end for stmt
   45
            return;
   46 }//end printLyrics function
   47
   48 int main()
   49 {
   50
           int numberOfRows; //the number of rows in the lyrics.
   51
           char music[MAX LINES][MAX LENGTH]; //the lyrics
   52
   53
           numberOfRows = readLyrics(music);
   54
           printLyrics(music, numberOfRows);
   55
   56
          printf("\n\n");
   57
           system("PAUSE");
   58
          return 0:
   59 }//end main function
```



Practice Problems

1. Write a program that will read in three lines of text from the keyboard (put the lines into a 2dimensional array). Once the lines are in the array, process the strings so that for each character in the alphabet you record the number of times that character appeared in total in the three lines of text. The output of the program should show the total number of times each letter appeared for all 26 letters in the alphabet.

