String Functions

partial list

Last updated 6/22/23

These slides introduce some standard C string functions

String Functions

- printf / scanf
  - printf()

```
printf("my string is: %s", myString);
```

Note: printf allows strings to be printed by name – no need to cycle through the elements

  - scanf()

```
char month[10];                  // create string
scanf("%9s", month);             // read in 9 characters
                                 // for the string called
                                 // month (adds the \0)
                                 // → 10 characters total

fflush(stdin);                   // required to remove
                                 // any extra characters
                                 // and the newline
```

** if we read in more characters than the string can hold we will overwrite unrelated data – don't forget 1 is used for the terminator

String Functions

- gets()
  - Get string – converts a line (up to newline or end-of-file) to a string
  - Prototype

    char * gets(char * stringPtr)

    Stores the string in stringPtr

    Returns stringPtr

```
char myString[81];          // standard 80 character line
                            // must be big enough to hold
                            // your line + delimiter
…
gets(myString);             // read one line of input
                            // typically don't use the return value
```

Remember: the name is a pointer

String Functions

- puts()
  - Put string – converts a string to a line of output (including the newline)
  - Prototype

    int puts(const char * stringPtr)

    Outputs the string + newline to stdout

    Returns a non-negative value if it worked, EOF if failed

```
char myString[22];              // some string
…
puts(myString);                 // output 1 line with value
                                // myString
```

String Functions

- strlen - #include <string.h>
  - String length – outputs the length of a string excluding the null character
  - Prototype

    size_t strlen(const char * string)

    Returns the length of the string

```
char myString[22];              // some string
…
foo = strlen(myString);
```

String Functions

- strcpy/strncpy - #include <string.h>
  - String copy – copy one string to another
  - Prototype

    char * strcpy(char * toStr, const char * fromStr)

    Copies fromStr to toStr

    Returns the address of toStr

    …

    strcpy(string2, string1);

    NO Boundary or Size checking is done

  - Use strncpy – only copies the size(N) number of characters

    char * strncpy(char * toStr, const char * fromStr, int size)

String Functions

- strcmp/strncmp - #include <string.h>
  - String compare – compare 2 strings
  - Prototype

    int strcmp(const char * str1, const char * str2)

    returns 0 if equal

    returns <0 if str1 < str2

    returns >0 if str1 > str2

    Note: compares ascii values

```
if(strcmp(mystr1, mystr2) == 0)
   …
```

  - Use strncmp - compare the first size(N) elements

    int strncmp(const char * str1, const char * str2, int size)

String Functions

- strcat/strncat - #include <string.h>
  - String concatenation – concatenate 2 strings
  - Prototype

    char * strcat(char * str1, const char * str2)

    Concatenates str2 onto str1 with result in str1

    Returns the address of str1

  ...

  strcat(stringA, stringB); // result in stringA

  NO Boundary or Size checking is done

  str1 must be large enough to hold the result

  - Use strncat – concatenate size(N) characters

    char * strncat(char * str1, const char * str2, int size)

String Functions

- Example

```
////////////////////////////////////////////////
/*
 * strings.c
 *
 *  Created on: Jan 23, 2018
 *      Author: johnsontimoj
 */

#include <stdio.h>
#include <string.h>

int main(void){
    setbuf(stdout, NULL);  // disable buffering

    int i;
    char st1[8] = "string1";
    char st2[8] = "string2";
    char st3[8];
    char st4[8];

    printf("st1 = %s\n", st1);
    printf("st2 = %s\n", st2);

    printf("st1 is made up of: ");
    for(i = 0; i < 8; i++){
        printf("- %c -", st1[i]);
    }
    printf("\n");

    printf("st1 is made up of: ");
    for(i = 0; i < 8; i++){
        printf("- %i -", st1[i]);
    }
    printf("\n");

    printf("enter a value for st3: ");
    scanf("%8s", st3);
    fflush(stdin);
    printf("you entered: ");
    printf("%s\n", st3);

    printf("enter a value for st4: ");
    gets(st4);
    printf("you entered: ");
    puts(st4);

    printf("the length of st1 is: %i\n", strlen(st1));

    strcpy(st4, st1);
    printf("st4 now = %s\n", st4);

    int foo;
    foo = strcmp(st1, st4);
    printf("foo = %i\n", foo);
    foo = strcmp(st1, st3);
    printf("foo = %i\n", foo);
    foo = strcmp(st2, st1);
    printf("foo = %i\n", foo);

    foo = strncmp(st1, st2, 6);
    printf("foo = %i\n", foo);

    char stA[15] = "";
    strcat(stA, st1);
    strcat(stA, st2);
    printf("%s\n", stA);

    return 0;
}// end main
```

Prints whole string

print as char

print as int

Problems | Tasks | Console | Properties

<terminated> (exit value: 0) Cons_Project.exe [C/C++ Application] D:\GDrive\MSOE\18_Q2_

```
st1 = string1
st2 = string2
st1 is made up of: - s -- t -- r -- i -- n -- g -- 1 --  -
st1 is made up of: - 115 -- 116 -- 114 -- 105 -- 110 -- 103 -- 49 -- 0 -
enter a value for st3: string3
you entered: string3
enter a value for st4: string4
you entered: string4
the length of st1 is: 7
st4 now = string1
foo = 0
foo = -1
foo = 1
foo = 0
string1string2
```

print terminator as char

print terminator as int