CPS 188

Computer Programming Fundamentals

Prof. Alex Ufkes

Topic 8.2: More string functions, examples

Toronto Metropolitan University

Notice!

Obligatory copyright notice in the age of digital delivery and online classrooms:

Today

Strings

More string functions

String examples

Strings

A **string** is a **character array** that is terminated by the ***null character*** – '\0'

```
#include <stdio.h>
int main()
{
    char city[] = {'T','o','r','o','n','t','o','\0'};
    printf("%d\n", '\0'); /* 0 on the asci table */
    return 0;
}
```

NUL ^@ 0	SOH ^A 1	STX ^B 2	E
DLE ^P	DC1 ^Q	DC2 ^R	D

String Initialization

```
#include <stdio.h>
int main()
{
    char city[] = “Toronto”;
    return 0;
}
```

When we initialize this way, the null character is automatically added to the end of the **char** array.

String size equals the number of characters plus one.

'\0'

```
int i;
char city[] = “Toronto”;

for (i = 0; city[i] != ‘\0’; i++)
{
     if (city[i] == ‘o’)
     {
          city[i] = ‘a’;
     }
}

printf(“%s”, city);
```

- Iterate until we hit the null character.
- Works on any string

Console:

Taranta

String Functions

```
#include <string.h>
```

String Functions

```
#include <string.h>

char city[] = “Oslo”;
int length = strlen(city);
printf(“Length: %d\n”, length);
```

Console

```
Length: 4
```

Length does not include the null character!

String Copy

```
char city1[8] = “Toronto”;
char city2[8];
city2 = city1;
```

ILLEGAL!

Use `strcpy` instead

strcpy

Found in `string.h`

```
char city1[8] = "Toronto";
char city2[8];

strcpy(city2, city1)
```

Copies string **city1** into string **city2**.

String Copy

```
char city[8];
city = "Markham";
```

Also ILLEGAL!

A string can be initialized **ONLY** upon declaration. Otherwise….

Use `strcpy` instead

strcpy

Found in `string.h`

```
char city[8];
strcpy(city, "Markham");
```

Copies string literal **"Markham"** into **city**.

strcpy copies *right* string into *left* string

More String Functions: `strncpy`

```
char *strncpy(char *dest, const char *src, size_t n)
```

Like **`strcpy`**, but only copies n characters.

`dest` Pointer to the destination string.

`src`
- Pointer to the source string.
- **const?** can't modify it. Strncpy won't modify src.

`n`
- Number of characters to copy.
- **size_t?** Alias for unsigned integer.

Returns a pointer to the copied string (`dest`)

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

int main (void)
{
  char str[] = "The quick brown fox jumped over the lazy dog";
  char dst[64];

  strncpy(dst, str, 19);

  puts(str);
  puts(dst);

  return (0);
}
```

Make sure **dst** is large enough!

Copy first 19 characters

Print both strings

strncmp.c

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

int main (void)
{
    char str[] = "The quick brown fox jumped over the lazy dog";
    char dst[64];

    strncpy(dst, str, 19);

    puts(str);
    puts(dst);

    return (0);
}
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
The quick brown fox jumped over the lazy dog
The quick brown fox♫o☻


------------------
(program exited with code: 0)

Press any key to continue . . .
```

What happened here…?

strncpy does not insert the null character!

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

int main (void)
{
  char str[] = "The quick brown fox jumped over the lazy dog";
  char dst[64];

  strncpy(dst, str, 19);
  dst[19] = '\0';

  puts(str); puts(dst);

  return (0);
}
```

We must do it ourselves

strncmp.c

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

int main (void)
{
    char str[] = "The quick brown fox jumped over the lazy dog";
    char dst[64];

    strncpy(dst, str, 19);
    dst[19] = '\0';

    puts(str);
    puts(dst);

    return (0);
}
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
The quick brown fox jumped over the lazy dog
The quick brown fox



------------------
(program exited with code: 0)

Press any key to continue . . .
```

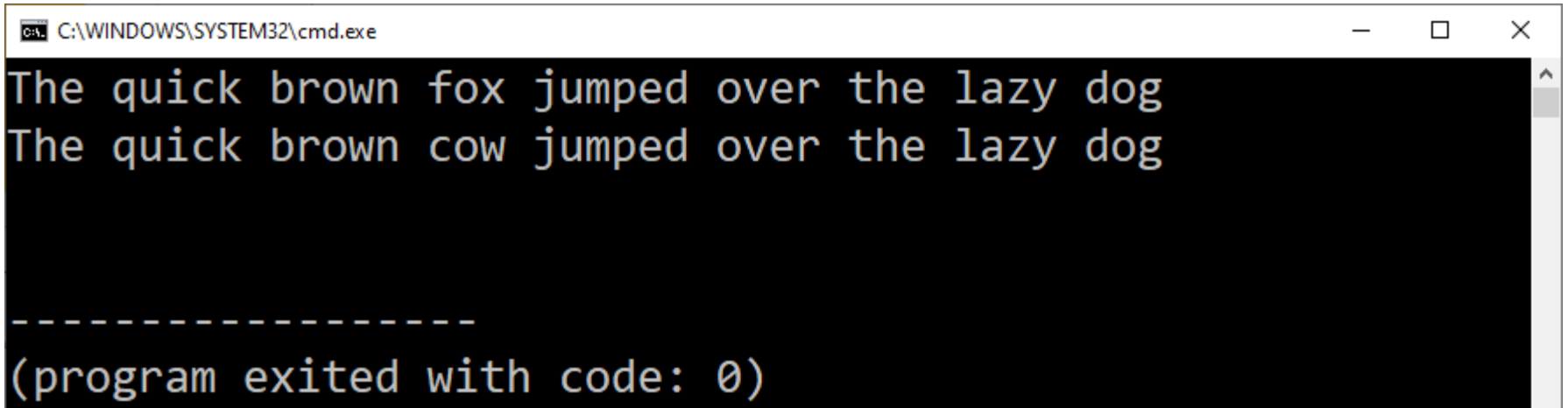
Example

Using string operations and functions, write a program that replaces “fox” with “cow” in the following string:

```
"The quick brown fox jumped over the lazy dog"
```

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

int main (void)
{
   char str[] = "The quick brown fox jumped over the lazy dog";
   char sub[] = "cow";

   puts(str);

   str[16] = ‘c’;
   str[17] = 'o’;
   str[18] = ‘w';

   puts(str);

   return (0);
}
```

- Hardcode individual character replacements?
- Works, but is clunky



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

int main (void)
{
   char str[] = "The quick brown fox jumped over the lazy dog";
   char sub[] = "cow";

   puts(str);

   strncpy(&str[16], sub, 3);

   puts(str);

   return (0);
}
```

- Strncpy into the interior of the destination string!
- Avoid individual character assignments.

Address of index 16, this is where ‘fox’ starts

str_examples.c

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

int main (void)
{
    char str[] = "The quick brown fox jumped over the lazy dog";
    char sub[] = "cow";

    puts(str);

    strncpy(&str[16], sub, 3);

    puts(str);

    return (0);
}
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
The quick brown fox jumped over the lazy dog
The quick brown cow jumped over the lazy dog



------------------
(program exited with code: 0)
```

More String Functions: `strcat`

```
char *strcat(char *dest, const char *src)
```

Concatenates (joins) two strings:

`dest`	Pointer to the destination string.
`src`	Pointer to the source string.

Appends the **`src`** string to the **`dest`** string.

Returns a pointer to the joined string (`dest`)

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

int main (void)
{
    char s1[64] = "Hello";
    char s2[] = ", World!";

    puts(s1);
    puts(s2);

    strcat(s1, s2);

    puts(s1);

    return (0);
}
```

- Once again, make sure dest string is large enough
- We've allocated 64 characters
- Only used six (Hello + \0)

strncat.c

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

int main (void)
{
    char s1[64] = "Hello";
    char s2[] = ", World!";

    puts(s1);
    puts(s2);

    strcat(s1, s2);

    puts(s1);

    return (0);
}
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
Hello
, World!
Hello, World!



------------------
(program exited with code: 0)

Press any key to continue . . .
```

More String Functions: `strncat`

```
char *strncat(char *dest, const char *src, size_t n)
```

Appends the first **n** characters of **`src`** to **`dest`**

- **strncpy** does NOT null terminate…
- But **strncat** DOES.
- We don't have to worry about adding the null character.

strncat.c

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

int main (void)
{
    char s1[64] = "Quick brown fox ";
    char s2[] = "jumped over the lazy dog";

    puts(s1);
    puts(s2);

    strncat(s1, s2, 6);

    puts(s1);

    return (0);
}
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
Quick brown fox
jumped over the lazy dog
Quick brown fox jumped


------------------
(program exited with code: 0)

Press any key to continue . . .
```

More String Functions: `strcmp`

This works:

```
2 < 7        3.14 >= 2.345        'a' == 'A'
```

This does not:

```
"Hello" < "World        "A" >= "B"
```

How can we compare strings? How do we normally do it?

More String Functions: `strcmp`

```
int strcmp(const char *s1, const char *s2)
```

Compares two strings *lexicographically* (by ASCII values)

`s1`	Pointer to the s1.
`s2`	Pointer to the s2.

Return value:

0 if strings are equal
>0 if first non-matching character in s1 is greater than in s2
<0 if first non-matching character in s1 is less than in s2

strcmp.c

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

int main (void)
{
    char s1[] = "cat", s2[] = "bat";
    char s3[] = "ban", s4[] = "con";

    printf("%d\n", strcmp(s1, s1));
    printf("%d\n", strcmp(s1, s2));
    printf("%d\n", strcmp(s2, s3));
    printf("%d\n", strcmp(s3, s4));
    printf("%d\n", strcmp(s4, s1));

    return (0);
}
```

C:\WINDOWS\SYSTEM32\cmd.exe

```
0
1
1
-1
1


------------------
(program exited with code: 0)

Press any key to continue . . .
```

More String Functions: `strcmp`

Want a fun challenge?

- Modify one of the sorting algorithms we saw to work on an array of strings
- **`strcmp`** and **`strcpy`** will come in handy!

gets, puts –VS– fgets, fputs

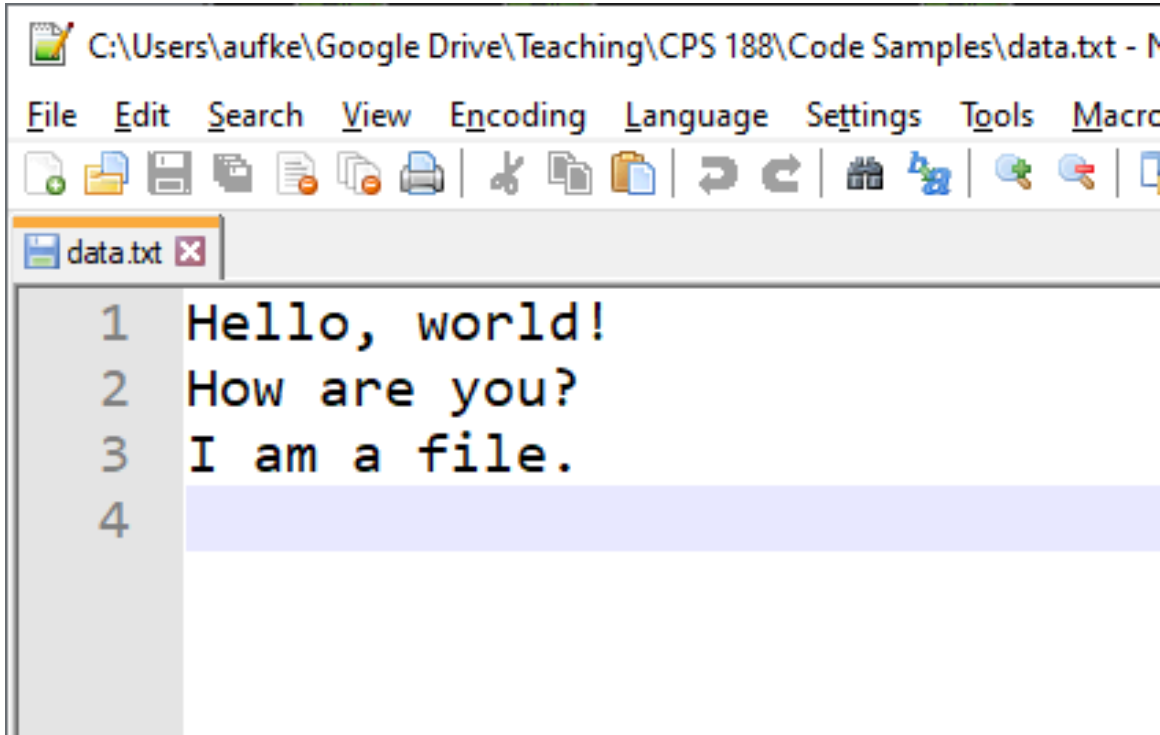
```
#include <stdio.h>

int main()
{
        char city[32];

        gets(city);
        puts(city);

        return 0;
}
```

- **gets** reads a string from stdin
- **stdin?** By default, the keyboard
- Think of **stdin** as a special file

- Similarly, **puts** prints a string to **stdout**
- **stdout?** By default, the terminal
- Think of **stdout** as a special file

fgets and **fputs** work similarly, but we can specify a different file (other than stdin/stdout)

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

int main (void)
{
    FILE *data = fopen("data.txt", "w");

    fputs("Hello, world!", data);
    fputs("How are you?", data);
    fputs("I am a file.", data);

    fclose(data);

    return (0);
}
```



fgetsputs.c

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

int main (void)
{
    //FILE *data = fopen("data.txt", "w");

    fputs("Hello, world!\n", stdout);
    fputs("How are you?\n", stdout);
    fputs("I am a file.\n", stdout);

    //fclose(data);

    return (0);
}
```

- Instead of a file, just use stdout!
- Works for fprintf as well.
- We don't need separate functions for file I/O and user I/O

C:\WINDOWS\SYSTEM32\cmd.exe

```
Hello, world!
How are you?
I am a file.


------------------
(program exited with code: 0)

Press any key to continue . . .
```

fgetsputs.c

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

int main (void)
{
    FILE *data = fopen("data.txt", "r");
    char line[64];

    fgets(line, 64, data);
    fputs(line, stdout);
    fgets(line, 64, data);
    fputs(line, stdout);
    fgets(line, 64, data);
    fputs(line, stdout);

    fclose(data);

    return (0);
}
```

- **fgets** used to read from a file. Newline delimited.
- It's a bit safer, we are required to pass the length of the destination string.

C:\WINDOWS\SYSTEM32\cmd.exe

```
Hello, world!
How are you?
I am a file.



--------------------
(program exited with code: 0)

Press any key to continue . . .
```

getc, putc –VS– getchar, putchar

```
char getchar()
```

- Get the next character from the standard input source (stdin).
- takes no arguments, returns the character as its result.

```
char getc(FILE *src)
```

- Get a single character from a file.
- Comparable to getchar, except that the character returned is obtained from a file (argument is a file variable).

getc, putc –VS– getchar, putchar

```
putchar(char ch)
```

- Prints a character to stdout. Terminal by default.

```
putc(char ch, FILE *dst)
```

- Prints a character to file dst.
- Can be stdout

getcputc.c

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

int main (void)
{
    putchar('H');
    putchar('e');
    putchar('l');
    putchar('l');
    putchar('o');
    putchar('\n');

    fputs("World", stdout);

    putc('!', stdout);
    putc('\n', stdout);

    return (0);
}
```

putchar(char ch)

- Prints a character to stdout.

putc(char ch, FILE *dst)

- Prints a character to file dst.
- Can be stdout

C:\WINDOWS\SYSTEM32\cmd.exe

```
Hello
World!


------------------
(program exited with c

Press any key to conti
```

More Character Functions: `ctype.h`

TABLE 8.3 Character Classification and Conversion Facilities in ctype Library

Facility	Checks	Example
`isalpha`	if argument is a letter of the alphabet	`if (isalpha(ch))` `  printf("%c is a letter\n", ch);`
`isdigit`	if argument is one of the ten decimal digits	`dec_digit = isdigit(ch);`

`islower`
`(isupper)`
	if argument is a lowercase (or uppercase) letter of the alphabet	`if (islower(fst_let)) {`
		`  printf("\nError: sentence ");`
		`  printf("should begin with a ");`
		`  printf("capital letter.\n");`
		`}`

`ispunct`	if argument is a punctuation character, that is, a noncontrol character that is not a space, a letter of the alphabet, or a digit	`if (ispunct(ch))`
		`  printf("Punctuation mark: %c\n",`
		`  ch);`

`isspace`	if argument is a whitespace character such as a space, a newline, or a tab	`c = getchar();`
		`while (isspace(c) && c != EOF)`
		`  c = getchar();`

Facility	**Converts**	**Example**

`tolower`
`(toupper)`
	its lowercase (or uppercase) letter argument to the uppercase (or lowercase) equivalent and returns this equivalent as the value of the call	`if (islower(ch))`
		`  printf("Capital %c = %c\n",`
		`  ch, toupper(ch));`

GNUPlot

- C does not provide standard libraries for plotting and graphics as part of the language.
- Some graphic libraries like OpenGL and WinBGIM can be used but are platform dependent and not portable.
- An external universal application named GNUPlot can be used for plotting in C. It is compatible with many platforms and languages.
- *Note: Plotting will not be on the final exam but you will need to do simple plots for the term project.*

GNUPlot Download

Download latest version here (5.4.6 as of today):

https://sourceforge.net/projects/gnuplot/files/gnuplot/5.4.6/

gnuplot Files

A portable, multi-platform, command-line driven graphing utility

Brought to you by: broeker, cgaylord, lhecking, sfeam

Summary | Files | Reviews | Support | Tickets ▾ | gnuplot-main | Mailing Lists | Discussion | News | Code

Download Latest Version
gp546-win64-mingw.exe (48.2 MB)

Get Updates

Home / gnuplot / 5.4.6

Name	Modified	Size	Downloads / Week
Parent folder			
readme	2023-02-12	907 Bytes	56
gp546-win64-mingw.exe	2023-02-12	48.2 MB	3,811
gp546-win64-mingw.7z	2023-02-11	45.6 MB	106
gnuplot-5.4.6.tar.gz	2023-02-11	5.7 MB	1,325
Gnuplot_5_4.pdf	2023-02-11	2.2 MB	80
RELEASE_NOTES_5_4_6	2023-02-11	15.8 kB	14
Totals: 6 Items		101.7 MB	5,392

gnuplot

File Plot Expressions Functions General Axes Chart Styles 3D Help

```
        G N U P L O T
        Version 5.4 patchlevel 6    last modified 2023-02-09

        Copyright (C) 1986-1993, 1998, 2004, 2007-2023
        Thomas Williams, Colin Kelley and many others

        gnuplot home:     http://www.gnuplot.info
        faq, bugs, etc:   type "help FAQ"
        immediate help:   type "help"  (plot window: hit 'h')

Terminal type is now 'qt'
gnuplot>
```

encoding: cp1252

Short and sweet tutorial for getting started:

http://personalpages.to.infn.it/~mignone/Numerical_Algorithms/gnuplot.pdf

Additional tutorials and documentation:

https://www.ap.smu.ca/~thacker/teaching/3437/gnuplot.pdf

https://web.physics.utah.edu/~detar/phys6720/resources/Gnuplot_tutorial.html

http://www.gnuplot.info/docs_5.4/Gnuplot_5_4.pdf

https://riptutorial.com/gnuplot

Questions?