String Manipulation

CST 357/457 – Systems Programming
Michael Ruth, Ph.D.
Associate Professor
Computer Science & I.T.
mruth@roosevelt.edu

Objectives

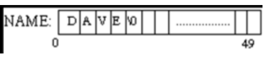
- Discuss string processing including comparisons, searching, & tokenizing
- Explain character testing & conversions

What to do with text once entered?

- There are multiple libraries in place for basic string processing
  - <string.h> - basic string functions
  - <ctype.h> - character testing
  - <stdlib.h> - string conversions

  - Remember from earlier discussions (arrays) that there is NO built-in type for strings
    - They are simply arrays of chars for ex:

[redacted]



String handling

- You can develop algorithms and methods to dice and slice your arrays but…
- You don't really want to reinvent the wheel, do you?

String Comparisons

- `int strcmp(const char *string1,const char *string2)`
- `int strncmp(const char *string1, char *string2, size_t n)`
- `int strcasecmp(const char *s1, const char *s2)`
- `int strncasecmp(const char *s1, const char *s2, int n)`

String comparison returns an int

- functions ***lexically*** compare the two input strings and return:
  - **Less than zero**
    - if string1 is lexically less than string2
  - **Zero**
    - if string1 and string2 are lexically equal
  - **Greater than zero**
    - if string1 is lexically greater than string2

More string functions

- Notice that almost all of the functions here take as arguments pointers to strings?
- This is so that they can handle variable lengths
- **`int strlen(const char *string)`**
  - Returns the number of chars in string

String Searching

- **`char *strchr(const char *string, int c)`**
  - Find first occurrence of character c in string
- **`char *strrchr(const char *string, int c)`**
  - Find last occurrence of character c in string
- **`char *strstr(const char *s1, const char *s2)`**
  - locates the first occurrence of the string s2 in string s1

String tokenizer in C?

- **`char *strtok(char *string, const char *delimiters)`**
  - break the string pointed to by string into a sequence of tokens,
    - each of which is delimited by one or more characters from the string pointed to by delimiters

Tokenizer Example

```
main() {

   char line[256];
   char *token;

   while(gets(line) != NULL)  {
      token = strtok(line, " ");
      while (token != NULL) {
         puts(token);
         token = strtok(NULL, " ");
      }
   }
}
```

Other functions of <string.h>

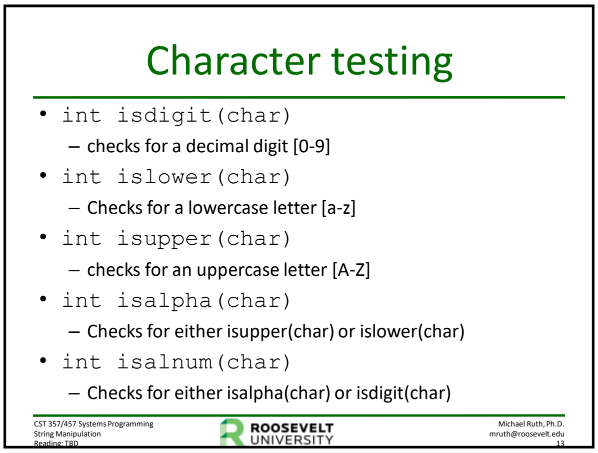
- `char *strerror( int errornum );`
  - Creates a system-dependent error message based on errornum
  - Returns a pointer to the string

```
int main() {

   int i;
   for (i=0;i<50;i++) {
      printf("%s \n",strerror(i));
   }
}
```

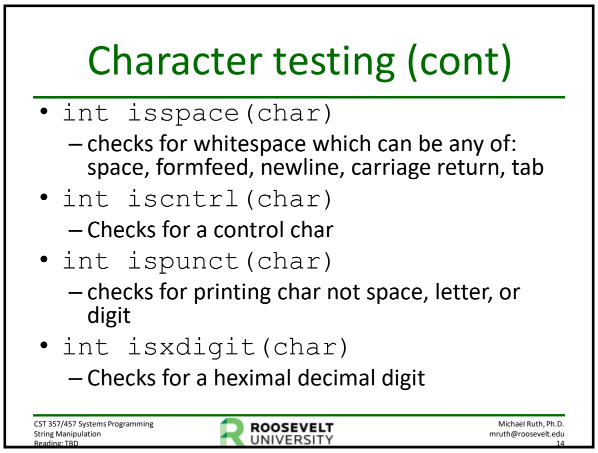
#include <ctype.h>

- Library used for character testing:
  - All of these functions take a character (or int) and return a non-zero int for true and zero for false
  - Quick ex:
    - `0 = isdigit('a')` 🡐 a is not a digit



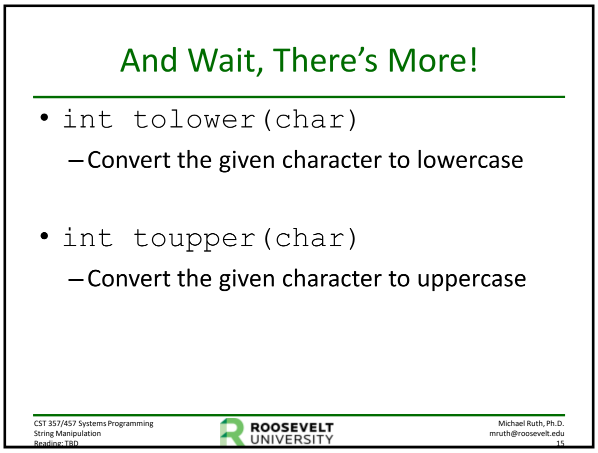
Character testing

- `int isdigit(char)`
  - checks for a decimal digit [0-9]
- `int islower(char)`
  - Checks for a lowercase letter [a-z]
- `int isupper(char)`
  - checks for an uppercase letter [A-Z]
- `int isalpha(char)`
  - Checks for either isupper(char) or islower(char)
- `int isalnum(char)`
  - Checks for either isalpha(char) or isdigit(char)



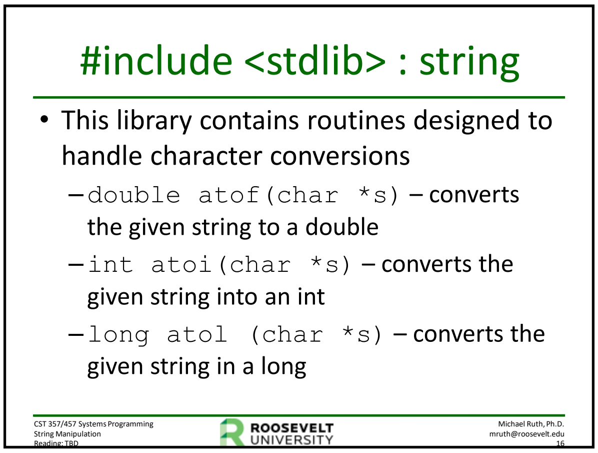
Character testing (cont)

- `int isspace(char)`
  - checks for whitespace which can be any of: space, formfeed, newline, carriage return, tab
- `int iscntrl(char)`
  - Checks for a control char
- `int ispunct(char)`
  - checks for printing char not space, letter, or digit
- `int isxdigit(char)`
  - Checks for a heximal decimal digit



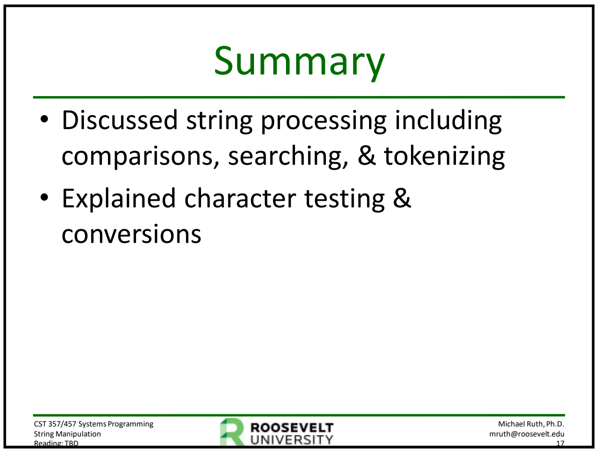
And Wait, There’s More!

- `int tolower(char)`
  - Convert the given character to lowercase
- `int toupper(char)`
  - Convert the given character to uppercase



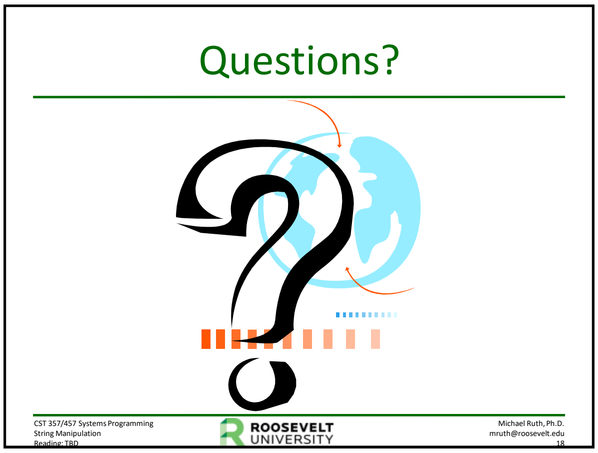
#include <stdlib> : string

- This library contains routines designed to handle character conversions
  - `double atof(char *s)` – converts the given string to a double
  - `int atoi(char *s)` – converts the given string into an int
  - `long atol (char *s)` – converts the given string in a long



Summary

- Discussed string processing including comparisons, searching, & tokenizing
- Explained character testing & conversions



Questions?