Character Testing

- The C++ library provides several functions for testing characters.
- Requires the cctype header file
- These functions have this signature:
  - `int isupper (int c);`
- They take a char argument
- They return non-zero for true, 0 for false.

```cpp
char input;
...
if (isupper(input)) ...
```
Character Testing

- isalpha      true for letter of the alphabet
- isalnum     true for letter or digit
- isdigit     true for digit
- islower     true for lowercase letter
- isprint     true for printable char
- ispunct     true for not (digit, letter or space)
- isupper     true for uppercase letter
- isspace     true for space, tab, newline

Case conversion

- int toupper (int c)
  - converts lowercase letters to uppercase
  - otherwise returns c
- int tolower (int c)
  - converts uppercase letters to lowercase
  - otherwise returns c
- Does NOT change argument

```c++
char x = 'A';
char y = tolower(x);
cout << x << " " << y << endl;
```
Output: A a
Internal Storage of Strings

- a C-string is
  - a sequence of characters
  - stored in consecutive memory locations
  - terminated by a null character (‘\0’)

- String: generic term for a sequence of characters.
- String literal: “this is a literal”
  - these are stored as C-strings in memory

- Programs store C-strings in char arrays.
- Do not need to pass size to functions taking C-strings as args, because the null char marks the end.

Library Functions for C-Strings

- These require the cstring header be included.

- String Length:
  - int `strlen` ( const char * str )
  - Returns number of characters in string (up to but not including the null char).
  - argument can be:
    - name of array containing a C-string
    - pointer variable holding address of a C-string
Library Functions for C-Strings

• String Concatenation:

  char * strcat ( char * destination, const char * source );

  Appends a copy of the source string to the destination string
  - destination must be long enough
  - returns destination, modified (can ignore)

  example:

  char string1[13] = “Hello “;
  strcat(string1, string2);
  cout << string1 << endl;

  Output: Hello World!

• String Copy:

  char * strcpy (char * destination, const char * source );

  Copies source string to destination
  - destination must be long enough
  - returns destination, modified (can ignore)

  example:

  char string1[13] = “Hello “;
  strcpy(string1, string2);
  cout << string1 << endl;

  Output: World!
Library Functions for C-Strings

• String Compare:
  • int \textbf{strcmp} ( const char * str1, const char * str2 );
  • Compares \texttt{str1} and \texttt{str2}
    - if \texttt{str1} and \texttt{str2} are the same, return 0
    - if \texttt{str2} comes after \texttt{str1} alphabetically, return -1
    - if \texttt{str2} comes before \texttt{str1} alphabetically, return 1
  • example:
    
    ```c
    char string1[13] = "Hello ";
    char string2[7] = "World!";
    if (strcmp(string1, string2) < 0)
      cout << "Negative" << endl;
    ```
    Output: Negative

• Substring:
  • char *\textbf{strstr} ( const char * str1, const char * str2 );
  • Returns a pointer to the first occurrence of \texttt{str2} in \texttt{str1}
    - returns a null pointer if \texttt{str2} is not part of \texttt{str1}
    - does not try to match terminating null char
  • example:
    
    ```c
    char str[] ="This is a simple string";
    char * pch;
    pch = strstr (str,"simple");
    strcpy (pch,"sample string");
    cout << str << endl;  
    ```
    Output: This is a sample string