Current Assignments

- Homework 5 will be available tomorrow and is due on Sunday.

  Arrays and Pointers
- Project 2 due tonight by midnight.
- Exam 2 on Monday. Review on Thursday.

  Functions (overloading, pass-by-value, pass-by-reference)

  Recursion

  Arrays

  Pointers

  Sorting (insertion sort, selection sort, and bubble sort)

Today

- Recursion from Homework 4
  - Recursive Adder
- Basic String Processing
- Recursion and Arrays
  - Palindrome
  - Recursive Insertion Sort

Basic String Processing

- Strings (arrays of chars) are very common
- They are usually referred to as char*s (remember a pointer is really the same as an array)
- …but be careful. Declaring an array allocates memory to store the data in, declaring a pointer just allocates memory for one address.

Fundamentals of Characters and Strings

- Character constant
  - –Integer value represented as character in single quotes
  - – `'z'` is integer value of `z`
    - `122` in ASCII

Fundamentals of Characters and Strings

- String - series of characters treated as single unit
  - Can include letters, digits, +, -, *, etc.

  - String literal (string constants)
    Enclosed in double quotes, for example: **"I like C++“**

  - Array of characters, ends with null character **'\0'**
  - String constant is a **const** pointer that points to string’s first character
    - Like arrays

Fundamentals of Characters and Strings

- String assignment
  - Character array
    - **`char color[] = "blue";`**
      - Creates 5 element **`char`** array **`color`** last element is **`'\0'`**
  - Variable of type **`char *`**
    - **`char *colorPtr = "blue";`**
      - Creates pointer **`colorPtr`** to letter **`b`** in string **“`blue`”**
        - » **“`blue`”** somewhere in memory
  - Alternative for character array
    - **`char color[] = { ‘b’, ‘l’, ‘u’, ‘e’, ‘\0’ };`**

Fundamentals of Characters and Strings

- Reading strings
  - Assign input to character array **`word[20]`**

```
cin >> word
```

    - Reads characters until whitespace or EOF
    - String could exceed array size

```
cin >> setw( 20 ) >> word;
```

    - Reads 19 characters (space reserved for `'\0'`)

String Manipulation Functions of the String-handling Library

- String handling library **`<cstring>`** provides functions to
  - Manipulate string data
  - Compare strings
  - Search strings for characters and other strings
  - Tokenize strings (separate strings into logical pieces)

`char *strcpy( char *s1, const char *s2 );`	Copies the string **s2** into the character array **s1**. The value of **s1** is returned.
`char *strncpy( char *s1, const char *s2, size_t n );`	Copies at most **n** characters of the string **s2** into the character array **s1**. The value of **s1** is returned.
`char *strcat( char *s1, const char *s2 );`	Appends the string **s2** to the string **s1**. The first character of **s2** overwrites the terminating null character of **s1**. The value of **s1** is returned.
`char *strncat( char *s1, const char *s2, size_t n );`	Appends at most **n** characters of string **s2** to string **s1**. The first character of **s2** overwrites the terminating null character of **s1**. The value of **s1** is returned.
`int strcmp( const char *s1, const char *s2 );`	Compares the string **s1** with the string **s2**. The function returns a value of zero, less than zero or greater than zero if **s1** is equal to, less than or greater than **s2**, respectively.

`int strncmp( const char *s1, const char *s2, size_t n );`	Compares up to **n** characters of the string **s1** with the string **s2**. The function returns zero, less than zero or greater than zero if **s1** is equal to, less than or greater than **s2**, respectively.
`char *strtok( char *s1, const char *s2 );`	A sequence of calls to **strtok** breaks string **s1** into "tokens"—logical pieces such as words in a line of text—delimited by characters contained in string **s2**. The first call contains **s1** as the first argument, and subsequent calls to continue tokenizing the same string contain **NULL** as the first argument. A pointer to the current to-ken is returned by each call. If there are no more tokens when the function is called, **NULL** is returned.
`int strlen( const char *s );`	Determines the length of string **s**. The number of characters preceding the terminating null character is returned.

String Manipulation Functions of the String-handling Library

- Copying strings
  - **`char *strcpy( char *s1, const char *s2 )`**
    - Copies second argument into first argument
      - First argument must be large enough to store string and terminating null character
  - **`char *strncpy( char *s1, const char *s2, size_t n )`**
    - Specifies number of characters to be copied from string into array
    - Does not necessarily copy terminating null character

String Manipulation Functions

- Concatenating strings
  - **`char *strcat( char *s1, const char *s2 )`**
    - Appends second argument to first argument
    - First character of second argument replaces null character terminating first argument
    - Ensure first argument large enough to store concatenated result and null character
  - **`char *strncat( char *s1, const char *s2, int n )`**
    - Appends specified number of characters from second argument to first argument
    - Appends terminating null character to result

String Manipulation Functions

- Comparing strings
  - Characters represented as numeric codes
  - Strings compared using numeric codes
  - Character codes / character sets
    - ASCII
      - "American Standard Code for Information Interchange"

String Manipulation Functions

- Comparing strings
  - **`int strcmp( const char *s1, const char *s2 )`**
    - Compares character by character
    - Returns
      - Zero if strings equal
      - Negative value if first string less than second string
      - Positive value if first string greater than second string
  - **`int strncmp( const char *s1, const char *s2, int n )`**
    - Compares up to specified number of characters
    - Stops comparing if reaches null character in one of arguments

String Manipulation Functions

- Tokenizing
  - Breaking strings into tokens, separated by delimiting characters
  - Tokens usually logical units, such as words (separated by spaces)
  - **`"This is my string"`** has 4 word tokens (separated by spaces)

String Manipulation Functions

- **char *strtok(char *s1, const char *s2)**
  - Multiple calls required
    - First call contains two arguments, string to be tokenized and string containing delimiting characters
      - »Finds next delimiting character and replaces with null character
    - Subsequent calls continue tokenizing
      - »Call with first argument **`NULL`**
- **Returns NULL if no characters matching the delimiter could be found**

String Manipulation Functions

```
/* strtok example */
#include <iostream>
#include <cstring>
 int main ()
{
   char str[] ="This is a sample string, just testing.";
   char * pch;
   cout << "Splitting string " << str << “into tokens: ”;
   pch = strtok (str," ");
   while (pch != NULL)
   {
     cout << pch;
     pch = strtok (NULL, " ,.");
    }
 return 0;
 }
```

String Manipulation Functions

Splitting string "This is a sample string, just testing."

in tokens:

This

is

a

sample

string

just

testing

String Manipulation Functions

- Determining string lengths
  - `int strlen( const char *s )`
    - Returns number of characters in string
      - Terminating null character not included in length

String Manipulation Functions

- Write iterative palindrome
- Write recursive palindrome
- Write recursive insertion sort