String Functions

Section 3.3

Introduction

- A number of operations can be performed on strings
- Some so common that C has functions for them
- To use, must include **string.h**
    - **#include <string.h>**

- Actually, quite a few string functions
- Only look at the most common ones
  - **strlen()** – string length
  - **strcmp()** – string compare
  - **strcpy()** – string copy
  - **strcat()** – string concatenate
  - **sprintf()** – formatted string

strlen()

- **strlen()** gives the length of a string
  - How many characters it has
  - **'\0'** is not counted in the length
    - strlen("Hello") is 5

```
int length;
char s[6];
s[0] = 'H';
s[1] = 'e';
s[2] = 'l';
s[3] = 'l';
s[4] = 'o';
s[5] = '\0';
```

```
length = 0;
while (s[length] != '\0')
{
    length++;
}
```

strcmp()

- **strcmp()** is for comparing two strings
    - Returns an integer indicating equal, less than, or greater than
        - **int result = strcmp(*string1*, *string2*);**

Strings	Results
"Hello" VS "Hello"	0
"Hello" VS "Hellp"	-1
"Hey" VS "Hallo"	1
"Hillo" VS "Hi"	1

```
int i;
int a;

char s[4];
s[0] = 'S';
s[1] = 'u';
s[2] = 'n';
s[3] = '\0';

char t[4];
t[0] = 'S';
t[1] = 'u';
t[2] = 'y';
t[3] = '\0';
```

```
i = 0;
a = 0;

while (a == 0)
{
    if (s[i] < t[i])
        a = -1;

    if (s[i] > t[i])
        a = 1;

    if (s[i] == '\0' || t[i] == '\0')
        break;

    i++;
}
```

strcpy()

- **strcpy()** is used to copy one string into another
  - Source string is second and destination string is first in arguments
  - **strcpy(*destination*, *source*);**

```
int i;
int a;

char s[4];
s[0] = 'S';
s[1] = 'u';
s[2] = 'n';
s[3] = '\0';

char t[4];
```

```
i = 0;

while (s[i] != '\0')
{
    t[i] = s[i];
    i++;
}

t[i] = '\0';
```

strcat()

- **strcat()** is used to concatenate a string to another string
  - Second argument is concatenated onto end of first argument and result put in first argument
  - **strcat(*string1*, *string2*);**
  - ***string1*** needs to be large enough to hold the result

```c
int i;
int j;

char s[6];
s[0] = 'H';
s[1] = 'e';
s[2] = 'l';
s[3] = '\0';

char t[3];
t[0] = 'l';
t[1] = 'o';
t[2] = '\0';
```

```c
i = strlen(s);
j = 0;


while (t[j] != '\0')
{
    s[i + j] = t[j];
    j++;
}


s[i + j] = '\0';
```

sprintf()

- **sprintf()** works like printf(), but instead of printing to the screen, it puts the formatted string in a string variable
    - Can use format specifiers like in printf()
    - **sprintf(*result*, "Amount is: $%.2f\n", money);**
        - Formatted string is stored in *result*

Example

- **string_example.c**
  - Asks for a string
  - Repeatedly asks for another string to compare with the first

Nonlibrary Problems

- Good to be familiar with familiar string library functions
- But, sometimes not a function to solve exactly the problem we have
- Need to also understand how strings work to be able to make our own string functions

Example

- Consider the problem of wanting to remove all occurrences of 'a' from a string
    - "Saturday" becomes "Sturdy"
- No library function to do exactly this
- Need to make our own

- **a_remover.c**
    - Uses two counters
    - Characters other than 'a' get copied over
    - All 'a's get removed

Converting Strings to Numbers

- Sometimes might need to convert a string containing a number to an actual number
- C provides some functions for doing such conversions
- Two common ones are **atoi()** and **atof()**
    - Both are included in stdlib.h

atoi()

- **atoi()** (a to i) converts a string of an integer into an actual integer
  - a is a letter, i is short for integer
- The input is a string containing an integer and the output is the actual integer value
  - **int x = atoi("1234");**

atof()

- **atof()** (a to f) converts a string of a real number into a double
  - a is a letter, f is short for floating point
- The input is a string containing a real number and the output is the corresponding double value
  - **double x = atof("3.1415");**