Chapter 8

C characters: In C a character occupies one byte. C uses ASCII.

We can store values from 0 to 255.

character	ASCII Code
'A'	65
'a'	97
'0'	48
space	32

The codes for other characters can be calculated if you remember these 4. For example, 'z' 122.

C is case sensitive. Sorting works by comparing ASCII codes.

"hello" "Hello"

Which comes first? Answer: "Hello"

Character Functions

There are functions for working with characters.

```
#include <ctype.h>

int isblank( int c );  // returns true if the parameter is a space


// Example of how to call the function

char c = 'x';
if (isblank(c))
   printf("%c is a blank\n", c);
```

Other functions in the ctype library are:

```
int isdigit( int c );
int isalpha( int c );
int isalnum( int c );
int isxdigit( int c );
   returns true if c is a valid hexadecimal digit
int islower( int c );
int isupper( int c );
int tolower( int c );
int toupper( int c );
int isspace( int c );
   true for space, tab, form feed, carriage return, vertical tab, newline
int iscntrl( int c );
   true if it is a control character
int ispunct( int c )
int isprint( int c )
   returns true if c is a printable character including space
int isgraph ( int c )
   true if c is printable, other than a space
```

C and null-terminated strings

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

In C a string is an array of characters. The end of the string is marked by a null character. '\0'

When you create a string, you need to think about the maximum number of characters you want to store. Then add 1.

Section 8.6.1 pg 351

```
#define SIZE1 25
#define SIZE2 15
```

These are called symbolic constants. Handled by the preprocessor (before your program is sent to the compiler). Every time it sees SIZE1 in your program, it replaces it by 25.

```
int main(void) {

   char x[ ] = "Happy Birthday to You";  // array of 22 character elements
   char y[ SIZE1 ]; // elements are garbage
   char z[ SIZE2 ]; // elements are garbage

   printf(" %s%s\n%s%s\n",
      "The string in array x is: ", x,
      "The string in array y is: ", strcpy( y, x ) );
          // strcpy function copies x to y
```

strcpy(x, y) would copy y to x. be careful about the order of the parameters

```
   // copy first 14 characters of x into z.  Doesn't copy the null character

   strncpy( z, x , SIZE2 - 1 );
   z[ SIZE2 - 1 ] = '\0';

   printf("The string in array z is : %s\n", z );

} // end main
```