MC140: lecture #14

today's topic:

global variables
logical operators
example from last time
2-dimensional arrays

global variables.

- variables must be declared before they are used
- we have used variables within main() and within functions
- *global variables*
  - declared outside main() and outside a function definition, usually at the top of the program, after # directives
  - can be "seen" anywhere

logical operators.

- in C, there are 3 logical operators:

meaning	C operator
NOT	!
AND	&&
OR	||

- they are used for complementary and complex truth expressions
  - where a simple truth expression is: `( z == 'Q' )`
  - complement is: `!( z == 'Q' )`
  - complex example: `( z == 'Q' ) || ( z == 'q' )`

example from last time.

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

// declare global variable
char *s = "The tendency of people to focus on the meaning of sentences influences their
ability to notice some of the obvious features."

// function prototypes
int issameletter( char c1, char c2 );
int countletter( char c, char *s );
int countwords( char * );
int countword( char *w, char *s );

int main( void ) {
   printf( "number of F's = %d\n",countletter( 'f', sent );
   printf( "number of words = %d\n",countwords( s ));
   printf( "number of THE's = %d\n",countword( "THE",s ));
} /* end of main() */
```



```
/* This function takes two character arguments,
   "c1" and "c2". It and returns true (1) if they
   are the same and false (0) if they are not the
   same. The function uses the ctype library
   function "tolower" to make the function
   insensitive to case. */

int issameletter( char c1, char c2 ) {
  return( tolower( c1 ) == tolower( c2 ) );
} /* end of issameletter() */
```



```
/* This function counts the number of occurrences of
   the argument character "c" inside the argument
   string "s". The function calls the function
   "issameletter()" to compare each character in "s"
   with "c". The function returns the number of
   occurrences of "c" in "s". The function returns 0
   if "c" is not found in "s". */

int countletter( char c, char *s ) {
  int i, count=0;
  for ( i=0; i<strlen( s ); i++ ) {
    if ( issameletter( s[i],c )) {
      count++;
    } /* end if */
  } /* end for i */
  return( count );
} /* end of countletter() */
```

```
/* This function counts the number of words in the argument
   string "s". The function uses a naïve algorithm, counting
   the number of spaces in the argument string and adding 1 to
   that count (to account for the last word in the string).
   This function will return an erroneous value if words are
   separated by more than one space and/or if the last word in
   the sentence is followed by one (or more) spaces. The
   function uses the ctype library function "isspace" to
   determine if a given character is a space. */

int countwords( char *s ) {
  int i, count = 0;
  for ( i=0; i<strlen( s ); i++ ) {
    if ( isspace( s[i] )) {
      count++;
    } /* end if */
  } /* end for i */
  return( count+1 );
} /* end of countwords() */
```



```
/* this function counts the number of occurrences of
   the argument word "w" in the argument string "s". */
int countword( char *w, char *s ) {
  int i, j, length_w, count = 0;
  length_w = strlen( w );
  for ( i=0; i<strlen( s )-length_w; i++ ) {
    j = 0;
    while (( j < length_w ) &&
           ( issameletter( s[i+j],w[j] ))) {
      j++;
    } /* end while j */
    if (( j == length_w ) && ( isspace( s[i+j] ))) {
      count++;
    } /* end if */
  } /* end for i */
  return( count );
} /* end of countword() */
```


arrays.

- recap:
  - elements: the related items
  - length: number of elements
  - index: position number of each element
  - example: `int x[25];`
- can be multi-dimensional
  - strings are one-dimensional char arrays
- a 2-dimensional array is called a *matrix*
  - example: `int y[5][5];`

pictures.

1-dimensional array declaration:
`int a[5]={-45,6,0,72,43};`

a[0]	-45
a[1]	6
a[2]	0
a[3]	72
a[4]	43

2-dimensional array declaration:
`int b[3][3]= {-45,6,0,72,43,1,4,19,-12};`

```
b[0][0] = -45
b[0][1] = 6
b[0][2] = 0

b[1][0] = 72
b[1][1] = 43
b[1][2] = 1

b[2][0] = 4
b[2][1] = 19
b[2][2] = -12
```

-45	6	0
72	43	1
4	19	-12

printing contents.

```
#include <stdio.h>
int main( void ) {
  int b[3][3]= {-45,6,0,72,43,1,4,19,-12};
  int i, j;
  for ( i=0; i<3; i++ ) {
      for ( j=0; j<3; j++ ) {
        printf( "b[%d][%d] = %d\n", i, j, b[i][j] );
      } /* end for j */
  } /* end for i */
} /* end of main() */
```


reading.

- material covered today:
  - DD: 4.10, 6.1-6.4, 6.9
- global variables are not treated as a separate topic in the textbook. we'll talk more about them when we discuss *scope*.