Lecture 3.3

Structures (structs)

What are structures?

- A structure is a complex data type built by using elements of other types. elements of other types.
- A structure provides a way of grouping variables under a single name for easier handling and identification.
- Structures may be copied to and assigned. They are also useful in passing groups of logically related data into functions.

How to declare a structure?

- A structure is declared by using the keyword **struct** followed by an optional structure tag followed by the body of the structure.

```
struct point {
   int x;
   int y;
};
```


How can I use this?

- The struct declaration is a user defined data type.
- This means that with a struct, YOU can, define YOUR OWN data types.
- Declaring
  - point left, right;
  - point origin;

is analogous to

  - float rate;

Using structures

- The individual members of a structure can be accessed using ".", the member access operator.
- Example

```
struct Dot {
    int x;
    int y;
};
.... ....
Dot location;
cout << "The x coordinate is "<< location.x;
cout << "The y coordinate is "<< location.y;
```


Nesting structures

- A rectangle could be represented as follows.

```
struct Rect{
    Dot upperLeft;
    Dot upperRight;
    Dot lowerLeft;
    Dot lowerRight;
};
Rect myRectangle;
```

- To access the members, just use the "dot" notation:

```
myRectangle.upperLeft.x=1;;
myRectangle.lowerRight.y=3;
```


Arrays of Structures

- A common way is to define an array of structs.
- Example,

  For counting the occurrence of each letter in a string.
- One approach: declare two separate arrays
  - One would hold the letters to compare against.
  - One to hold a count of the occurrences of each letter. letter.

```
#define NUMLETTERS 26
....
char letter[NUMLETTERS];
int count[NUMLETTERS];
```


Arrays of Structures (cont)

-

```
#define NUMLETTERS 26
struct lettertype {
    char letter;
    int count;
};
typedef lettertype Letter;
```

- The keyword **typedef** provides a mechanism for creating synonyms for previously defined data types.
- Letter alphabet[NUMLETTERS];

Arrays of structures (cont2)

- The previous declarations could also be written as follows.

```
typedef struct lettertype {
        char letter;
        int count;
} Letter;
Letter alphabet[NUMLETTERS];
```


Sample Program

```
#include <stdio.h>
#include <string.h>
#define NUMLETTERS 26
struct Letter {
    char letter;
    int count;
} ;
```


Sample Program (cont 2)

```
int main()
{
    int i,j;
    int length;
    Letter alphabet[] = {
        'a',0, 'b',0,'c',0, 'd',0,'e',0, 'f',0, 'g',0, 'h',0,
        'i',0,'j',0,'k',0, 'l',0, 'm',0, 'n',0, 'o',0,'p',0,
        'q',0, 'r',0,'s',0,'t',0, 'u',0, 'v',0,
        'w',0, 'x',0, 'y',0, 'z',0};
    char sampleString [] =
        "now is the time for all good men to come to the
        aid of their country";
```


Sample Program (cont 3)

- /* Find length of string */

```
length = strlen(sampleString);
for (i = 0; i < length; i++){
        for (j = 0; j < NUMLETTERS; j++) {
                if (sampleString[i] == alphabet[j].letter){
                        alphabet[j].count++;
                }
        }
}
```

Sample Program (cont 4)

```
for (j = 0; j < NUMLETTERS; j++) {
   printf("%c found %d times\n",
        alphabet[j].letter, alphabet[j].count);
   }
   return 0;
}
```