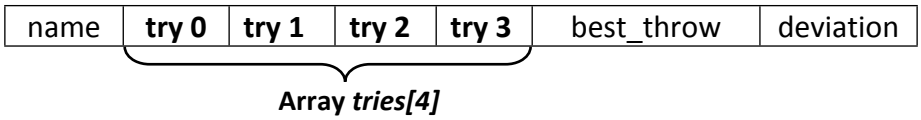
PROBLEM: This program uses structures and pointers.
It will receive (via a file) a set of javelin tries for a set of throwers. We will figure out the best throw for each competitor, the average of the best throws, and the winning best throw.
You have to write function ***get_stats*** and add to a ***makefile***.

(**1**) First in **lab8.h,** you need to declare a structure type ***thrower_t***.
I named my structure ***thrower_t*** and its 4 parts are: //help on C9 slide9
a character array ***name*** that is 20 in length (0-19),
a double array of ***tries*** that is N_TRIES in length,
a double named ***best_throw***, and // refers to individual's best throw.
a double named ***deviation***.



(**2**) Next in **lab8.h**, you need to declare a structure type ***stats_t***. //help on C9 slide9
I named my structure ***stats_t*** and its 2 parts are:
variables, all type double, named: ***average_of_best_throws***, ***winning_throw*** .
// winning_throw refers to the all-over best throw

(3) In lab8.h, you need to add your name in the comment block. *When the program is working* ***correctly***, you will need to shift the comment marks [//] on the four #define statements
At the start, it looks like this:

```
//#define OUT_FILENAME "lab8.txt"
#define OUT_FILENAME "lab8sample.txt"

//#define IN_FILENAME "lab8.dat"
#define IN_FILENAME "lab8sample.dat"
```

At the end, it needs to look like this:

```
#define OUT_FILENAME "lab8.txt"
//#define OUT_FILENAME "lab8sample.out"

#define IN_FILENAME "lab8.dat"
//#define IN_FILENAME "lab8sample.dat"
```

(**4**) Write the function *get_stats*. It is started in a file for you. The prototype is:

```
void get_stats(thrower_t throw_list[NCOMPETITORS],   /* in & out */
               stats_t *throw_stats);                /* output   */
```

Its job will be to figure out the best throw for each thrower, compute the all-over average of all the best throws, find the winning throw, and calculate each thrower's deviation from the winning throw. There is a file already started with the first line enclosed.

(5) You will be provided a test driver program and related functions that need some changing. You will only need to add:

- your name in twice in the function *open_out_file.c*
- the two structures you will add to *lab8.h*
- the function *get_stats* which you have to write in its own file
- need to shift from the sample data file to the real data file in *lab8.h* once it all works with the sample data.
- Add to the *makefile* the necessary additions to make it work with *get_stats.c*

FILES TO COPY:
First move to your class folder by typing: **cd csc60**
The following command will create a directory named **lab8** and put all the needed files into it below your csc60 directory.
Type: **cp -R /home/college/bielr/files_csc60/lab8 .**
Spaces needed: (1) After the **cp** ↑ *Don't miss the space & dot.*
(2) After the **-R**
(3) After the directory name at the end & before the dot.
After the files are in your account and you are still in **csc60**,
you need to type: **chmod 755 lab8**
This will give permissions to the directory.
Next move into lab8 directory (**cd lab8)**, and
type: **chmod 644 *.***
This will give permissions to the files.
Your new lab8 directory should now contain:
lab8.c, lab8.h, lab8.dat, lab8sample.dat, get_data.c, open_out_file.c, print_all.c, a starting file for *get_stats.c,* and add to the *makefile* so it can deal with *get_stats.c*

INPUT/OUTPUT DESCRIPTION:
The program input is a set of thrower's names and their four tries with the javelin. The lengths are type double.
Each record/line of the file has a thrower's name and four tries, and space for the best_throw of the thrower and the deviation from the all-over winning jump.
The file consists of the four names, followed by the first four tries, and then each successive set of tries.

REMINDERS: Check the validity of your answers.

- You need a **makefile** for this assignment.

- If a parameter is passed into a function without the asterisk, you are to use the "dot" notation.

- If the parameter is passed into a function with the asterisk, you are to use the "points into"

notation (->).

- Here is a helpful list of the full variable names, for your use.

```
throw_list[r].name
throw_list[r].tries[c]
throw_list[r].best_throw
throw_list[r].deviation
throw_stats->average_of_best_throws
throw_stats->winning_throw
```

- If using the sample data file, the output is going to *lab8sample.txt*
- If using the final data file, the output is going to *lab8.txt*

- The print_all function does all the printing and ***is a model for referencing the variables***.

NOTE: Apparently in Java, long names are sometimes put into variables with shorter names. In C this does not save time. If you try it, be sure to run the sample data until you get correct answers.

ALGORITHM DEVELOPMENT - Pseudocode:

```
/*-------------------------------------------------------------*/
main                        //provided
   out_file = open_out_file ();
   get_data(IN_FILENAME, throw_list);
   get_stats(throw_list, &throw_stats);
   print_all(out_file, throw_list, &throw_stats);

/*-------------------------------------------------------------*/
FILE * open_out_file(void)     //provided
   Open the output file
   Return the output file pointer.

/*-------------------------------------------------------------*/
                                  //Provided
void get_data (char *filename,                          /* input  */
          thrower_t throw_list[NCOMPETITORS] );   /* output */
  Open the appropriate file
  Read the data into the throw_list array
        (Use double loops, one each for columns and rows)
  Close the file
/*-------------------------------------------------------------*/
/*                                  Provided              */
void print_all(FILE * out_file,   thrower_t throw_list[NCOMPETITORS] , stats_t *throw_stats t)

        /* This routine does all the printing and is a model for referencing the variables. */
/*-------------------------------------------------------------------*/
```

```
/*    This is a sub-function that you must write */

void get_stats( thrower_t throw_list[NCOMPETITORS],   /* in & out */
                stats_t *throw_stats )                /* in & out */
{
     Zero out the average_of_best.  (HINT: use the -> notation)
     Zero out the winning_throw.

     loop from r=zero to r< NCOMPETITORS, increment by one
     {
          set the thrower's best_throw to the thrower's first throw
          loop from c=one to c< N_TRIES, increment by one
          {
               if the next throw  >  the contents of the best throw column
               {
                    set the best column to this better throw try
               }
          }          /* end of the loop using "c" */
          add the thrower's best throw into the running total average_of_best or sum

          figure the winning-best throw of all the throws (use an IF {}  )

     }               /* end of the loop using "r"  */

     compute the average of the best throws

     loop from r=zero to < NCOMPETITORS increment by one
    {
          figure the thrower's deviation from the winning_throw
          (deviation is all-over winning_throw minus each thrower's best throw)
    }               /* end of the second loop using "r"  */
    return
}
/*-------------------------------------------------------------*/
```

NOTE: To correct indentation in *vim*, type: :1 then type: =G

HAND EXAMPLE:
This is a sample example. This sample data is in the file named *lab8sample.dat*. This output goes to a file named *lab8sample.txt*

```
Your Name, Lab 8 output.

Track Results

Name              Try 1   Try 2   Try 3   Try 4  Best Throw  Deviation
---------------   -----   -----   -----   -----  ---------   ---------
Jayne Johnson     40.10   29.00   55.50   54.67     55.50      19.50
Missy Monroe      51.30   30.50   26.20   34.56     51.30      23.70
Nell Niner        62.70   37.80   37.40   67.12     67.12       7.88
Lanny Loop        17.00   75.00   48.60   69.58     75.00       0.00


Best Throw Average  =  62.23 meters

Winning Throw       =  75.00 meters
```

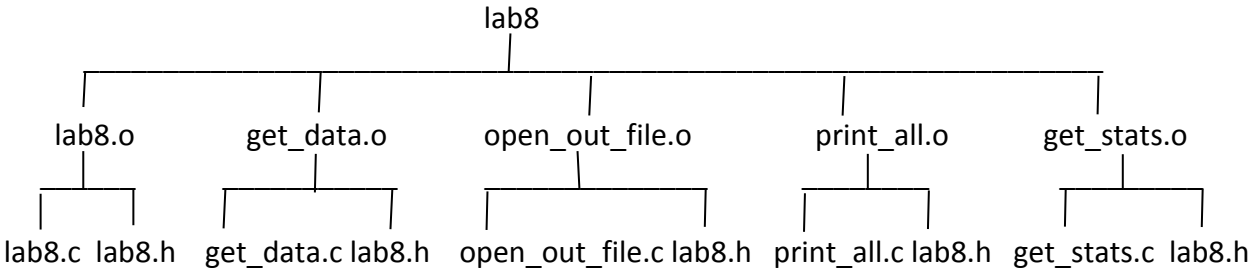
Comments:

In function *get_data*, I had a lot of trouble getting the columns to line up. When I used a conversion specifier of %20c, the names printed out, but each line of numbers moved one space to the left. Then I tried %s. The string had a New-Line buried in it, resulting in a name on one line, and its numbers on the next line.

So finally, I read the string using "fgets", function-get-string. The string still contains the New-Line, but I also used the function "strchr" to search the string for the location of the New-Line, and changed that position in the string to a NULL. Everything then printed out OK.
(One can do a "man fgets" for more information.)

A string, by definition, is a character array where the last character is NULL.
That is why there is code in *get_data* to put a NULL into the last location of the string.

DEPENDENCY CHART



The makefile needs additions made for it to deal with the function get_stats.

→ **more on next page**

PREPARE YOUR FILE FOR GRADING:

*Make sure your program has been set to use **lab8.dat** and has been re-complied (gcc). Information on making that change is on Page1.part3 of these directions.*

When all is well and correct,

Type: **script StudentName_lab8.txt** [Script will keep a log of your session.]
Type: **touch lab8.h** to allow all the files to be recomplied
Type: **make** to compile & link the code
Type: ./**lab8** run the program (or **lab8** or whatever name you used in the makefile)
Type: **cat lab8.txt** to show contents of the final output file
Type: **exit** to leave the script session

DELIVERABLES: 25 points. Available 4/10. Due 4/25. Lock on 5/2

No Zip Files please.
Go to Canvas and turn in:

1. lab8.h
2. your script session (StudentName_lab8.txt)
3. makefile
4. get_stats.c
5. open_out_file.c