Monday 27 October

Class Activity 18:

Objectives:

Learn a little about similarities/differences between MATLAB and C

The following program, CircleArea, is an implementation of a C-program in a structured format. Individual tasks are separately designed and programmed as *subprograms* or *functions*. Note that the "main" subprogram, **`int main(void)`**, consists of a set of tasks to accomplish the program goals and is simply called **`main( )`**. We can read the **`main`** subprogram to get an idea of *what* is being done to accomplish the program goals. If we want to know *how* each of the tasks is performed, we need to find the subprogram corresponding to the task. Subprograms are separated by a comment line, **`//**************`**, as an aid in finding individual subprograms.

Activity: Transfer the file ca18a.c from the **`get`** directory to the **`temp`** directory. Enter the C++ environment and retrieve the file into the editor.

Activity: Use the arrow keys to look through the program. Note the structure of the main subprogram, called **`main( )`**. It consists of four tasks named for the actions which need to take place for the program to function properly. The first task (**`displayheader`**) runs only once. The other three tasks are inside a program repetition loop.

(a) Inside main, note that three variables are declared. What type is associated with each variable declaration? Do the types make sense for the variable?

(b) Note that the program repetition loop is a **`do..while`** loop. Because the exit condition is checked at the end of the loop, it does not need to be initialized in the same manner as a **`while`** loop.

(c) The first three commands in the program repetition loop are directed toward performing the purpose of the program (i.e., get radius, compute area, display results). How does the information flow from one command to another in this section?

(d) The last three commands in the program repetition loop are directed toward asking the user whether to repeat the calculations. **`fflush`** is a C-command that empties the keyboard input buffer, which is called **`stdin`**. Emptying the keyboard buffer before getting a character is always a wise idea when getting character information in C. **`fflush`** is found in the **`stdio.h`** library (which was included at the top of the program).

(e) Note the use of the **`toupper`** command in the **`while`** condition at the end of the **`do..while`** loop. What do you think this command does? Why use it? What is the alternative to not using it? **`toupper`** is found in the **`ctype.h`** library.

Activity: Run the program.

Activity: Modify the program to also compute the circumference. Use a "structured" approach similar to that already being used. What new subprogram needs to be created? Can you modify an existing one?

To accomplish this activity, pay attention to ***where*** and ***how*** names are declared and used in the original program. Work by analogy: If a similar variable name is used as declared in a statement, do likewise with your variable. If a variable name appears as **`*pname`** in a parameter list, you will want your new variable to do likewise. If the variable name appears as **`pname`** in the **`scanf`**, do likewise with your variable name. You need to use a similar "style" for the new variable **`circumference`**. Among the modifications, you will need to add a new subprogram prototype to the prototype list near the top of the program (what name seems appropriate?). You will also need to type in the new subprogram code, say after the subprogram **`double computearea`**. You will also need to declare a new variable (data object) in **`main`** (what name seems appropriate?). You will also need to add your new variable name to the parameter list in the prototype and definition for subprogram **`displayresults`**.

Activity: Working by analogy, create a new program, **`rectangle`**, to compute the area and perimeter of a rectangle.

For really simple programs, such as this one, the use of structured programming design techniques and subprograms is obviously overkill. However, the ideas behind structured programming design are most easily illustrated using such simple programs. Early formation of the habits of good programming practice will be extremely important as our programming complexity increases. Well-designed, structured programs are much easier to debug and verify than programs which have no discernable structure.

Recap: You should have learned

- Basic structure of a C program.

Turn in: Listing of your modified program to compute the circumference and area and screen display from running the program.

Class activity extra credit: Turn in a listing and screen display from running your rectangle program.

```
// program CircleArea
/*
  Your Name
  Your e-mail address
  Engineering 0012 - Introduction to Computing
  Class Activity ??
  Date Created:
  Date Last Update:
```

```
  this program determines the area of a circle of
  a radius entered by the user and displays the results
  to the screen
*/

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

// defined constants
#define  PI  3.1415926536  // value of pi

// data structures

// subprogram prototypes
  void  displayheader( void );
  void  getradius( double *pradius );
  double  computearea( double radius );
  void  displayresults( double radius, double area );

/*******************************************************************/
main( )
/*
   control strategy for program CircleArea
*/
{ // begin main
  // data object declaration
    double  radius;  // radius of circle
    double  area;  // area of circle
    char  goagain;  // program repetition flag

    displayheader();

  // algorithm
    do
    {
       getradius( &radius );
       area = computearea( radius );
       displayresults( radius, area );

       printf( "\n\nAnother area? Yes or No ==> " );
       fflush( stdin );
       scanf( "%c", &goagain );

    } while (toupper(goagain) == 'Y');

  return( 0 );

} // end main

//*******************************************************************
void  displayheader( void )
/*
  purpose: display output header
  goal state: output header displays in output window
*/
{ // begin displayheader

  // algorithm
    printf( "\nYour Name" );
    printf( "\nYour e-mail address" );
    printf( "\nEngineering 0012 - Introduction to Computing" );
    printf( "\nClass Activity ??\n" );

} // end displayheader

//********************************************************************
void  getradius( double *pradius )
/*
  purpose: obtain value of radius from user
  goal state: *pradius holds radius value
*/
{ // begin getradius
  // algorithm
    printf( "\n\nPlease enter value for radius.\n\nRadius ==> " );
    scanf( "%lf", pradius );

} // end GetRadius

//********************************************************************
double  computearea( double radius )
/*
  purpose: given radius, compute area of circle
  goal state: area returned
*/
{ // begin computearea

  // algorithm
    return( PI*radius*radius );

} // end computearea

//********************************************************************
void  displayresults( double radius, double area )
/*
  purpose: display radius and resulting area on screen
  goal state: radius and area displayed
*/
{ // begin displayresults

  // algorithm
    // output results to screen
      printf( "\nThe area of a circle of radius %lf is %lf\n\n",
                   radius, area );

} // end displayresults
```