EECS 10: Assignment 4

Prepared by: Che-Wei Chang, Prof. Rainer Dömer

July 14, 2014

$\boxed{\text{Due Monday July 21, 2014 11:00pm}}$

Based on the program `calculator.c` for assignment 3, you will be first asked to extend the floating-point calculator with several advanced functions:

- **option 6**: Get the approximate square root of the current result;
- **option 7**: Get the approximate value of Pi;
- **option 8**: Get the cosine of current result;
- **option 9**: Get the approximate Nth root of the current result (Extra credits);

More specifically, the menu will look like as the following for this assignment:

```
...
1. Add a floating point number to the current result;
2. Subtract a floating point number from the current result;
3. Multiply the current result by a floating point number;
4. Divide the current result by a floating point number;
5. Take the absolute value of the current result;
6. Get the approximate square root of the current result;
7. Get the approximate value of Pi;
8. Get the cosine of current result;
9. Get the approximate Nth root of the current result;
10. Quit

Please enter a selection:
```

Note: In this assignment, please make "Quit" as **option 10** (no matter you have the bonus or not).

1 Initial Setup

Before you start working on this assignment, please do the following steps:

1. Create the subdirectory *hw4* for this assignment, and change your current directory to *hw4*.
2. We will modify and extend the *calculator* program based on assignment 3. Please feel free to reuse all your source code (*.c) as the starting point for this assignment. You may reuse the solution files to assignment 3 which are published on our course website as well.
3. Copy the .c source file from your own account:

```
    zuma% cd hw4
    zuma% cp ../hw3/calculator.c ./
```

4. Or copy the .c source file from the *eecs22* account:

```
zuma% cd hw4
zuma% cp ~eecs10/hw4/calculator.c ./
```

2 Option 6: Square root approximation [20 points]

Extend the calculator with the $6th$ function to calculate the approximate square root of the current floating-point value.

2.1 Square root approximation algorithm

We will use a binary search approximation technique for this assignment. In particular, the program will always keep a range of a left bound L and a right bound R, where the actual square root S lies somewhere between L and R: $L \leq S \leq R$ Consequently, it follows that $L*L \leq N = S*S \leq R*R$. Thus, to find S, we can compare $L*L$ or $R*R$ with N. The binary approximation then works as follows. First, we compute a value M that lies in the middle between the left bound L and the right bound $R : M = L + (R-L)/2$. Then, if $M*M$ is less than N, the square root obviously lies somewhere in the right half of the current range (i.e. within M to R), otherwise in the left half of the current range (i.e. within L to M). The program then can use the proper half of the range as the new range and repeat the whole process. With each iteration, the search range is effectively reduced to half its previous size. Because of this, this technique is called binary search. To start the search, we will use the range from 0 to N (which is guaranteed to contain the square root of N). We will stop the iteration, once we have reached a range that is smaller than 0.00001 so that we reach a precision of 5 digits after the decimal point for our approximation.

The pseudo-code of the algorithm can be written as follows:

```
Start with a range of 0 to N
As long as the range is not accurate enough, repeat the following steps:
Compute the middle of the range
Compare the square of the middle value with N
If the middle value is less than the square root
              Use middle-to-right as the new range
Otherwise
              Use left-to-middle as the new range
Output the middle of the latest range as result
```

For example, to compute the square root of 10, the program will start with 5, which is in the middle between 0 and 10. Since $5*5 = 25$ is larger than 10, the program will try the middle number 2.5 of left bound (0 to 5). Thus, the program compares $2.5*2.5$ with 10. Because the result 6.25 is smaller than 10, it will pick 3.75 (the middle number of 2.5 and 5) as the next guess. By picking the middle number every time and comparing its square with the original number, the program gets closer to the actual square root.

To demonstrate the approximation procedure, your program should print the approximated square root in each iteration, as follows, if the user chooses ”6” as the option:

```
--------------------------------------------
The current result is: 10.000000
1. Add a floating point number to the current result;
2. Subtract a floating point number from the current result;
3. Multiply the current result by a floating point number;
4. Divide the current result by a floating point number;
5. Take the absolute value of the current result;
6. Get the approximate square root of the current result;
7. Get the approximate value of Pi;
8. Get the cosine of current result;
9. Get the approximate Nth root of the current result;
```

```
10. Quit

Please enter a selection: 6
--------------------------------------------
Iteration 1: the square root of 10.00000 is approximately 5.00000
Iteration 2: the square root of 10.00000 is approximately 2.50000
Iteration 3: the square root of 10.00000 is approximately 3.75000
Iteration 4: the square root of 10.00000 is approximately 3.12500
...
Iteration 20: the square root of 10.00000 is approximately 3.16228
```

Note that your program should run properly for any real number which is the current result value (not only for the demo value 10).

3 Option 9: Bonus Problem [5 points]

Improve your program with the 9-th option so that it can calculate the n-th root of any value. The value n should be a positive integer input by the user.

For example, your program should look like this for the bonus part:

```
--------------------------------------------
The current result is: 42.000000
1. Add a floating point number to the current result;
2. Subtract a floating point number from the current result;
3. Multiply the current result by a floating point number;
4. Divide the current result by a floating point number;
5. Take the absolute value of the current result;
6. Get the approximate square root of the current result;
7. Get the approximate value of Pi;
8. Get the cosine of current result;
9. Get the approximate Nth root of the current result;
10. Quit

Please enter a selection: 9
--------------------------------------------
Please input the value of integer n (n>0): 5
Iteration 1: the 5th root of 42.00000 is approximately 21.00000
Iteration 2: the 5th root of 42.00000 is approximately 10.50000
...
Iteration 23: the 5th root of 42.00000 is approximately 2.11179
```

Note that your program should run properly for any real number which is the current result value and any integer n which is greater than 0 (not only for the demo 5th root of value 42).

4 Option 7: Pi Approximation by using the Monte Carlo (MC) methods [20 points]

Extend the calculator with the $7th$ function to calculate the approximate value of Pi.

4.1 Monte Carlo approximation of Pi

Monte Carlo (MC) methods are stochastic techniques, meaning they are based on the use of random numbers and probability statistics to investigate problems. In this part of the homework, you are asked to write a program to implement a simple geometric MC experiment which calculates the value of Pi based on a "hit and miss" integration.

The figure below shows a unit circle circumscribed by a square. The radius of the circle r equals to 1/2 of the side of the square. Furthermore, the center of the circle and the center of the square are identical.

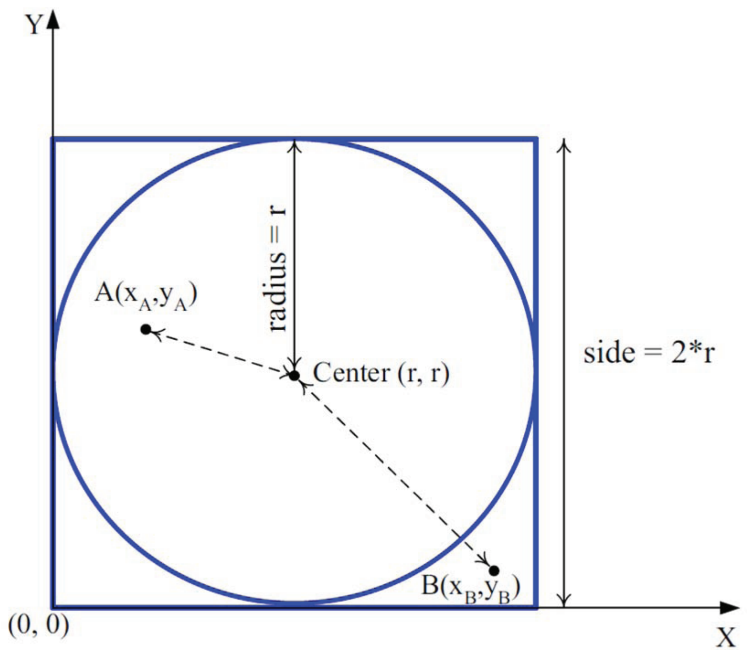

Figure 1: A Circle Circumscribed by a Square

Imagine we can throw points randomly at the above figure. If we can throw infinite random points, it should be apparent that of the total number of points that hit the circle divded by the the number of points that hit within the square is proportional to the area of that part.
In other words:

$$\frac{number\ of\ points\ hitting\ circle\ area}{number\ of\ points\ hitting\ square\ area} = \frac{area\ of\ circle}{area\ of\ square} = \frac{\pi \times r \times r}{(2 \times r)^2} = \frac{\pi \times r \times r}{4 \times r \times r} = \frac{\pi}{4}$$

Therefore, we get the formula to calculate π using Monte Carlo method:

$$\pi = 4 \times \frac{number\ of\ points\ hitting\ circle\ area}{number\ of\ points\ hitting\ square\ area}$$

In the real world, we can only throw a finite number of random points, therefore, the π calculated using the above formula is an approximation of the exact value of π.

We can have our computer generate random numbers to simulate the throwing of points. For each point, we can have computer to generate two random floating point numbers to be the x and y coordinates of the point, where $0 \leq x \leq 2r$ and $0 \leq y \leq 2r$ so that (x, y) must fall within the square area. However, this randomly generated point could fall within the circle area or fall out of the circle area.

To decide if the randomly generated point (x, y) is within the area of the circle, we can compare the distance of the point to the center with the radius r. For example, the point A in the above figure is in the circle area since its distance to center is less than r. However, the point B in the above figure is not in the circle area since its distance to center is greater than r.

Note: If the distance of point to the center equals to radius r, then that point is considered within the circle area.

Assume the radius of the circle is r and the coordinates of the randomly generated point P is (x, y), then the distance of P to the center is:
$Distance(P, Center) = \sqrt{(x-r) \times (x-r) + (y-r) \times (y-r)}$

To avoid the square root calculation, you can compare $Distance(P, Center) \times Distance(P, Center)$ with the radius squared $r \times r$ in order to decide if the randomly generated point is within the circle area.

Your program should look like the following for this part:

```
--------------------------------------------
The current result is: any number
1. Add a floating point number to the current result;
2. Subtract a floating point number from the current result;
3. Multiply the current result by a floating point number;
4. Divide the current result by a floating point number;
5. Take the absolute value of the current result;
6. Get the approximate square root of the current result;
7. Get the approximate value of Pi;
8. Get the cosine of current result;
9. Get the approximate Nth root of the current result;
10. Quit

Please enter a selection: 7
--------------------------------------------
Enter the radius of circle: 5
Enter the number of points: 10
Point No.1(x=8.401877,y=3.943829): IN
Point No.2(x=7.830992,y=7.984400): IN
Point No.3(x=9.116474,y=1.975514): OUT
Point No.4(x=3.352228,y=7.682296): IN
Point No.5(x=2.777747,y=5.539700): IN
Point No.6(x=4.773971,y=6.288709): IN
Point No.7(x=3.647845,y=5.134009): IN
Point No.8(x=9.522297,y=9.161951): OUT
Point No.9(x=6.357117,y=7.172969): IN
Point No.10(x=1.416026,y=6.069689): IN
/******In Summary******/
Points within circle area: 8
Points out of circle area: 2
Pi= 3.200000
```

More specifically, when the user chooses option 7, your program should ask for the input of radius r and the number of random points N the computer needs to generate. The output should like this:

```
Enter the radius of circle:  5
Enter the number of points:  10
```

During the generation, whenever a random point is generated, your program should print out the coordinates of the point and whether the point is *In* or *Out* the circle area like this:

```
Point No.1(x=0.000000,y=6.551714):  OUT
Point No.2(x=3.048189,y=6.749779):  IN
Point No.3(x=1.067537,y=5.165868):  IN
Point No.4(x=4.896695,y=6.024659):  IN
Point No.5(x=3.699454,y=2.566607):  IN
Point No.6(x=3.741874,y=8.255867):  IN
Point No.7(x=1.727042,y=2.977996):  IN
Point No.8(x=6.435438,y=7.896664):  IN
Point No.9(x=9.878231,y=8.005921):  OUT
Point No.10(x=4.642476,y=5.389874):  IN
```

At the end, your program should output the number of points within and out of the circle, together with the approximation value of π like this:

```
/******In Summary******/
Points within circle area:  8
Points out of circle area:  2
Pi= 3.200000
```

HINT In assignment 3, you have learned how to generate random integer numbers within the range of $[0,n)$ (n is exclusive). However, in this homework, you need to generate floating point numbers with the range of $[0,n]$ (n is inclusive). Therefore, there are some modification of the code described in assignment 3.

You need to replace the following line in assignment 3
randomNumber = rand() % n;
where *randomNumber* is an integer variable
with
*randomNumber = ((double)rand())/((double)RAND_MAX)*s;* /* s is the side of the square */
where *randomNumber* is a double variable

Furthermore, in assignment 4, we want you to use the same seed for the random numbers generation in order to generate the same series of random numbers. Therefore, take out the following line:
#include ⟨time.h⟩

and replace the following line in homework4
srand(time(NULL)); with
srand(0);

5 Option 8: Get the cosine of current result value [10 points]

Extend the calculator with the 8*th* operation to get the cosine of the current result value.

Please just call (use) the `cos()` function from `math.h` to get the cosine value.

Note: to compile a program using the math library, you need to add the ”-lm” option to gcc.

```
gcc calculator.c -o calculator -Wall -ansi -lm
```

6 Implementation

To implement the program, you need to define a set of more functions; each function does a specific job.

6.1 Function Declarations

The following function declarations are recommended:

```
/*Get the approximate square root return the value*/
double ApproximateSquareRoot(double x);

/*Get the approximate value of Pi and return the value*/
double ApproximatePi(void);

/*Get the approximate nth root return the value*/
double ApproximateNthRoot(double x, unsigned int n);
```

Functions `ApproximateSquareRoot()` requires one double as its input parameter which is the current result value. `ApproximateNthRoot()` requires one double and one integer as its input parameters which the first is the current result value and the second is the *n* for Nth root. `ApproximatePi()` calculate the approximate value of Pi and does not require any parameters.

6.2 Global Variables

As assignment 3, we can keep the same global variables , as follows:

```
double currValue;
double newValue;
```

7 What to submit

You need to save your program as *calculator.c*.

The way you demonstrate your code will depend on whether or not you completed the extra credit. If you did not complete the extra credit, perform the following steps to generate the script file:

1. Compile and run your program
2. Input the floating point number 4.2
3. Add the current result with 5.8
4. Get the square root of the current result value
5. Multiply the current result by 0
6. Get the approximate value of Pi with radius = 5, number of points = 30000
7. Get the cosine value for the current result
8. Exit your program

If you did complete the extra credit, perform the following steps to generate the script file:

1. Compile and run your program

2. Input the floating point number 4.2
3. Add the current result with 2.7
4. Get the square root of the current result value
5. Multiply the current result by 0
6. Add the current result with 28
7. Get the 5th root of the current result value
8. Get the approximate value of Pi with radius = 5, number of points = 20
9. Get the cosine value for the current result
10. Exit your program

For your reference, the result obtained after the last step should be -0.989992.

Name the script file *calculator.script*. If you forgot how to create a script file, check the instructions for assignment 1.

You also need to create a *calculator.txt* file, which briefly explains your program.

8 Submission

Submission for the files will be similar to last weeks assignment. Put all the files for assignment 4 in directory **hw4/** and run the **/ecelib/bin/turnin10** command in the parent directory of **hw4/** to submit your homework.

Note: please pay attention to any announcements on the course messageboard.