

CS 2213, Spring 2009
Assignment 2: A Single Function, Iterative Program (v2)
Due: 5pm, Friday, February 6, 2009

Goal: The purpose of this assignment is to learn to write relatively simple single-function C programs.

Objectives: be able to write single-function C programs making use of loops, double and integer variables; be able to write C programs making use of the functions: `printf()`, `rand()`, and `sqrt()`; be able to write C programs including various standard header files; and be able to compile programs making use of math functions (such as `sqrt()`).

Directions

Permissible Collaboration and Resources. *This assignment is to be completed individually.* You are allowed to discuss general strategies for solving assignments with fellow students or other individuals, but it is no longer a ‘general strategy’ if the discussion gets to the level of detail of what would be done in actual lines of code found in these programs and discussions of ‘general strategy’ should not be taking place while either party is editing their source code. (It is, however, perfectly acceptable to discuss C language constructs and concrete examples of them which are not taken directly from anyone’s solution to an assignment.) In addition, you may seek more specific help from mentors and lab tutors in trying to understand why their code doesn’t work. Furthermore, you may also seek help and guidance of any kind from the TA and instructor. *You may not look at any code for a Monte Carlo estimation of pi other than what is contained in this document or included in other material/links provided by the instructor or TA.*

The Square Root Program and the Math Library. There are several standard C functions that provide basic numerical operations for floating point numbers similar to what is available in the Math class in Java. The declarations of these functions are provided by the `<math.h>` header file.

Examine the provided `sqrt2.c` source file. It uses the function: `double sqrt(double x)`, which is declared in `math.h`. (Note the `#include` at the top of `sqrt2.c`.)

Try to compile `sqrt2.c`. What kind of error message do you get? This error indicates that, even though there was a declaration of `sqrt`, included from `math.h`, that `gcc` knows about as it compiles `main`, later, when `gcc` tries to ‘link’ everything together, `gcc` cannot find the definition (function body) for `sqrt()`. This is because the math functions (such as `sqrt()`) are in a separate library that you have to tell `gcc` to explicitly include. This is accomplished by simply adding the `-lm` switch to the `gcc` command-line, e.g.:

```
gcc -std=c99 -pedantic -Wall -Wextra -Wstrict-prototypes \  
    -o sqrt2 -lm sqrt2.c
```

Compile and run `sqrt2.c`.

Taking the Square Root of a Random Number. The C standard library provides a function `int rand(void)`, which returns a pseudo-random integer i where $0 \leq i \leq \text{RAND_MAX}$. Both the declaration for `rand()` and the preprocessor macro `RAND_MAX` can be obtained by including the header file `<stdlib.h>`.

If one needs an expression that results in a random floating point number from the closed interval $[0,1]$ instead of a random integer, one can simply cast the result to a double and divide that by `RAND_MAX`: `(double)rand()/RAND_MAX`. To get a random floating point number from $[0,5]$, one could use: `5.0 * rand() / RAND_MAX`. Why is a cast to `double` needed in the first expression above (for $[0,1]$) but not in the second expression (for $[0,5]$)?

Copy `sqrt2.c` to `sqrt-rand.c`. Modify `sqrt-rand` to obtain 10 pseudo-random floating point numbers in the interval $[5,10]$, and for each random number, print out a line containing the number and what its square root is, for example:

```
sqrt(9.200939) = 3.033305
```

Pay careful attention to the types of your variables.

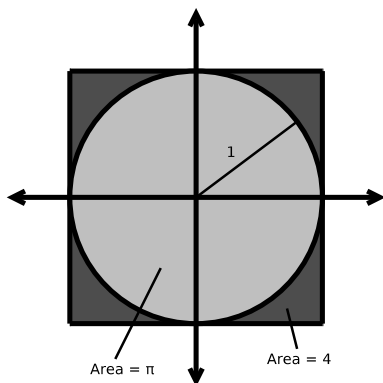
Compile and run your program. Check the results to see if the program is generating 10 different (non-integer) floating point numbers between 5 and 10, and check that it is computing the square root of those numbers correctly. (Hint: pick a couple of the numbers and use a calculator to square the square root.) If necessary, correct any bugs and repeat.

Monte Carlo Estimation of π (pi). One way to estimate the value of π is to use a Monte Carlo simulation of randomly choosing points within a square and then determining how many of those points fall within the circle inscribed in that square. If the process for randomly choosing points gives equal likelihood for each point within the square, then the probability of a chosen point being within the circle is determined by the ratio $\frac{\text{AREA}(\text{circle})}{\text{AREA}(\text{square})}$. If the circle is the unit circle (radius 1), the square will have sides of length 2, and the ratio will be:

$$\frac{\text{AREA}(\text{circle})}{\text{AREA}(\text{square})} = \frac{\pi r^2}{l^2} = \frac{\pi 1^2}{2^2} = \frac{\pi}{4}$$

which means:

$$\pi = 4 \frac{\text{AREA}(\text{circle})}{\text{AREA}(\text{square})}$$



If a lot of points are chosen randomly, then the ratio of the number of points within the circle to the total number of points will approximate $\frac{\pi}{4}$.

Here is an algorithm for performing this Monte Carlo estimation of π :

```
num_in_circle :=0

do COUNT times
    x := pick a random real number from [-1,1]
    y := pick a random real number from [-1,1]

    if  $\sqrt{x^2+y^2} < 1$  then
        increment num_in_circle
    end if
end do

pi := 4 *  $\frac{\text{num\_in\_circle}}{\text{COUNT}}$ 
print "pi is about ", pi, new_line
```

Implement this algorithm as a C Program. Use the function `rand()` as described above to generate the random numbers in the interval `[-1,1]`. Pay careful attention to which variables should be integer (those that are being used to count) and those which should be double (those that are being used to represent real numbers used in or resulting from calculations). Use a preprocessor macro to define the value of `COUNT` as `1000`.

Save your implementation as `pi.c`.

Compile, and run your program. What output does your program produce when compiled and run? If you don't get a results between 3.0 and 3.25, there's probably something wrong. Debug your program and repeat if necessary.

Modify the program to use a count of `1000000`. What result did you get? Is this result closer to the correct value of π ?

Submission. Please create a README file divided into three sections: ANSWERS TO QUESTIONS, NOTES TO GRADER, and CERTIFICATION.

1. Under the heading ANSWERS TO QUESTIONS. Please answer the following questions:

- What error message did you get when you compiled `sqrt2.c` without a `-lm`?
- Why is a cast to double needed in the expression `(double)rand() / RAND_MAX` but not needed in `5.0 * rand() / RAND_MAX`?
- What value did your pi program print out when `COUNT` was `1000`?
- What value did your pi program print out when `COUNT` was `1000000`?
- Is the value for `COUNT=1000000` more accurate?

2. Under NOTES TO GRADER, add any notes, you wish the TA/instructor to consider while grading.

In particular, please document any known problems with your program. If gcc produces any warning or error messages, please list them, describe what

you think they mean, and say whether you think it indicates a problem in your code or if gcc is just being 'too picky.'

3. Under CERTIFICATION, please answer the following two questions in your README file:

- (a) Did you use any code/materials other than those provided by or referenced by the instructor or TA in completing this assignment? If so, please cite the source (including its URL if it is online) and the nature and extent of your use of that material.

If such material is prohibited by the directions for this assignment, points may be deducted, but as long as such use is completely disclosed in the README file (including source, nature, and extent), it will not be considered to constitute scholastic dishonesty.

- (b) Did you work with anyone or get help in completing this assignment from anybody other than the instructor or TA? (This includes mentors and tutors.) If so, please describe the nature and extent of the help received.

If such help or collaboration exceeds what is allowed for this assignment according to the directions above, points may be deducted, but as long as the help or collaboration is completely disclosed in the README file, it will not be considered to constitute scholastic dishonesty.

Below your answer to those two questions, include the following statement:

I certify that except as noted above, the work I submit for this assignment has been completed solely by me without any outside help and without looking at any code for a Monte Carlo estimation of pi other than what is contained in this document or included in other material/links provided by the instructor or TA.

Below this statement, please write your name and date.

Use svn to commit your final submission by 5pm on the due date.

1. Make sure that you have all of the required files and they've all been added to svn. The expected contents of assgn1 directory in the repository includes: assignment2.pdf, sqrt2.c, README, sqrt-rand.c, and pi.c. It is a also good idea to view your repository with your web browser to make sure all of the files are there.
2. You can continue to make changes, and re-commit as often as you want. The last version committed before 5pm on the due date will be graded.