

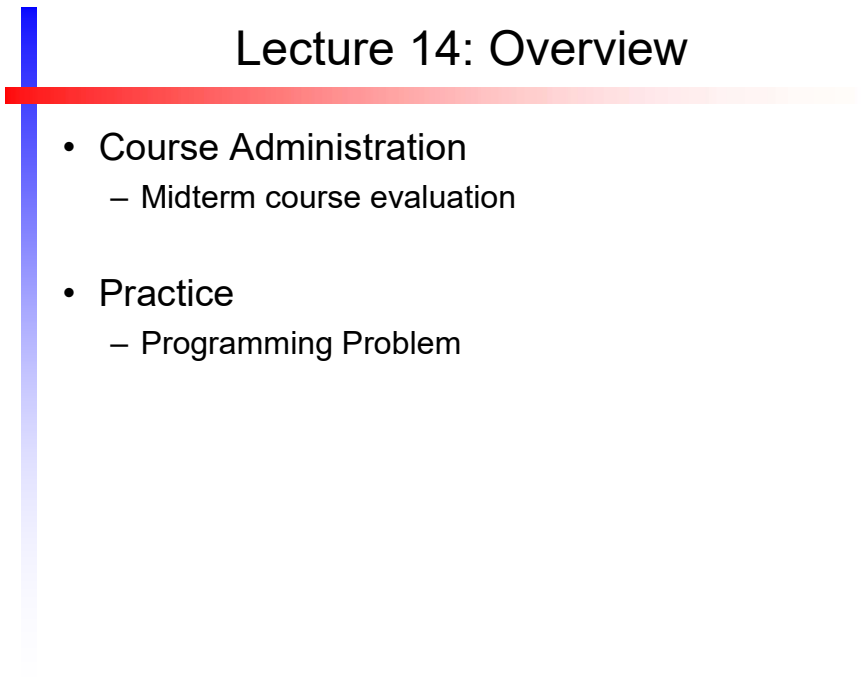
EECS 22: Advanced C Programming

Lecture 14

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine



Lecture 14: Overview

- Course Administration
 - Midterm course evaluation
- Practice
 - Programming Problem

EECS22: Advanced C Programming, Lecture 14 (c) 2017 R. Doemer 2

Course Administration

- Midterm Course Evaluation
 - This week!
 - Wednesday, Oct. 25, 8am – Nov. 1, 8am
 - Online via EEE Evaluation application
- Feedback from students to instructors
 - Completely voluntary
 - Completely anonymous
 - Very valuable
 - Help to improve this class!
- Mandatory Final Course Evaluation
 - expected for week 10 (TBA)

EECS22: Advanced C Programming, Lecture 14

(c) 2017 R. Doemer

3

Programming Problem

- Task:
 - Write a program that calculates the square root of a positive number entered by the user
- Instructions:
 - Write a main module (file `main.c`) that prompts the user for a value and prints the calculated square root
 - Write a square root module (files `sqrt.c` and `sqrt.h`) which implements a function with the signature `double SquareRoot(double)`
 - Write a corresponding `Makefile` to compile the program
- Algorithm:
 - Use a binary search algorithm to calculate the square root (see next page)

EECS22: Advanced C Programming, Lecture 14

(c) 2017 R. Doemer

4

Binary Search Algorithm For Square Root

- Square Root Approximation Algorithm:
 - Input: positive real number N
 - Output: square root of N
 - Approximate the square root by use of a range $\{L, R\}$, where $L \leq \text{sqrt}(N) \leq R$
 - Start with the range $\{0, \max(1, N)\}$
 - Calculate the middle of the range $M = L + (R-L)/2$
 - If the square root of N lies in the lower half of the range, use $\{L, M\}$ as new range; otherwise use $\{M, R\}$
 - Repeat the bisection until the range is smaller than $1 \cdot 10^{-5}$
 - Output M
- Hint:
 - $L \leq \text{sqrt}(N) \leq R \Leftrightarrow L^2 \leq N \leq R^2$

EECS22: Advanced C Programming, Lecture 14

(c) 2017 R. Doemer

5

Binary Search Algorithm For Square Root

- Example: **Makefile**

```
# Makefile:

SquareRoot: sqrt.o Main.o
    gcc sqrt.o Main.o -o SquareRoot

sqrt.o: sqrt.c sqrt.h
    gcc -c -Wall -ansi -std=c99 sqrt.c -o sqrt.o

Main.o: Main.c sqrt.h
    gcc -c -Wall -ansi -std=c99 Main.c -o Main.o

# EOF
```

EECS22: Advanced C Programming, Lecture 14

(c) 2017 R. Doemer

6

Binary Search Algorithm For Square Root

- Example: **Main.c**

```
/* Main.c: main program file */  
  
#include <stdio.h>  
#include "sqrt.h"  
  
int main(void)  
{  
    double x, s;  
  
    do{ printf("Enter a positive value: ");  
        scanf("%lf", &x);  
    } while(x < 0.0);  
    s = SquareRoot(x);  
  
    printf("The square root of %g is %g.\n", x, s);  
    return 0;  
} /* end of main */  
  
/* EOF Main.c */
```

Binary Search Algorithm For Square Root

- Example: **sqrt.h**

```
/* sqrt.h: header file for square root approximation */  
  
#ifndef SQRT_H  
#define SQRT_H  
  
double SquareRoot(double n);  
  
#endif /* SQRT_H */  
  
/* EOF sqrt.h */
```

Binary Search Algorithm For Square Root

- Example: `sqrt.c`

```
/* sqrt.c: square root approximation */
#include <assert.h>
#include "sqrt.h"

double SquareRoot(double n)
{ double l, r, m;

  assert(n >= 0.0);
  l = 0;
  r = (n > 1.0) ? n : 1.0;
  do { m = l + (r-l)/2;
      if (n < m*m)
        { r = m; }
      else
        { l = m; }
    } while(r-l < 1e-5);
  return m;
}
/* EOF sqrt.c */
```