EECS 10: Computational Methods in Electrical and Computer Engineering

Lecture 13

Rainer Dömer

doemer@uci.edu

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

Lecture 13: Overview

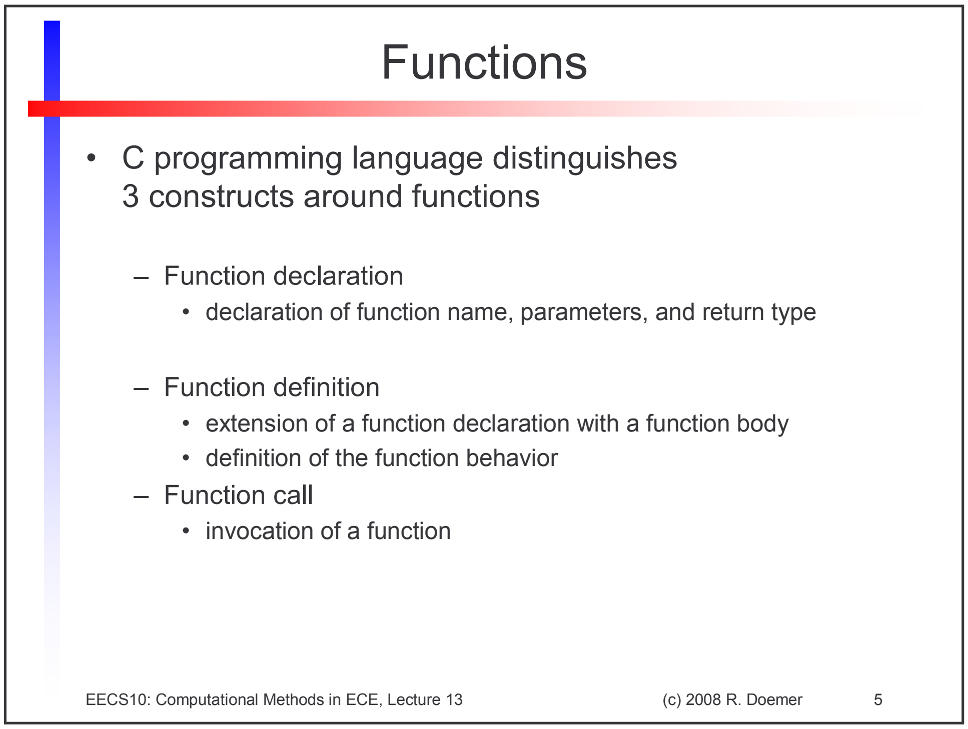
- Course Administration
  - Midterm course evaluation
- Functions
  - Introduction to function concepts
    - Function declaration
    - Function definition
    - Function call
  - Simple functions
    - Example **`Square.c`**
  - Hierarchy of functions
    - Example **`Cylinder.c`**

Course Administration

- Midterm Course Evaluation
  - Six days, starting today!
  - Oct. 29, 2008, 8am - Nov. 3, 2008, 8pm
  - 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)

Functions

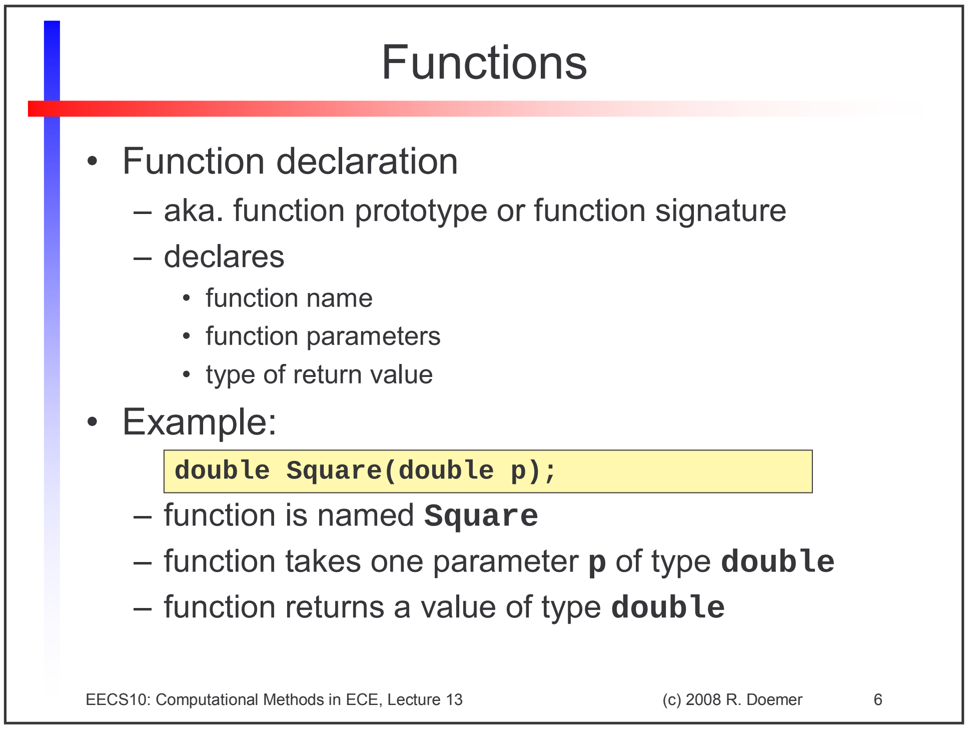
- Introduction to Functions
  - Important programming concepts
    - Hierarchy
    - Encapsulation
    - Information hiding
    - Divide and conquer
  - Software reuse
    - Don’t re-invent the wheel!
  - Program composition
    - C program = Set of functions
      - starting point: function named `main`
    - Libraries = Set of functions
      - predefined functions (often written by somebody else)



Functions

- C programming language distinguishes 3 constructs around functions
  - Function declaration
    - declaration of function name, parameters, and return type
  - Function definition
    - extension of a function declaration with a function body
    - definition of the function behavior
  - Function call
    - invocation of a function

EECS10: Computational Methods in ECE, Lecture 13 (c) 2008 R. Doemer 5



Functions

- Function declaration
  - aka. function prototype or function signature
  - declares
    - function name
    - function parameters
    - type of return value
- Example:

```
double Square(double p);
```

  - function is named **`Square`**
  - function takes one parameter **`p`** of type **`double`**
  - function returns a value of type **`double`**

EECS10: Computational Methods in ECE, Lecture 13 (c) 2008 R. Doemer 6

Functions

- Function definition
  - extends a function declaration with a function body
  - defines the statements executed by the function
  - may use local variables for the computation
  - returns result value via **`return`** statement (if any)
- Example:

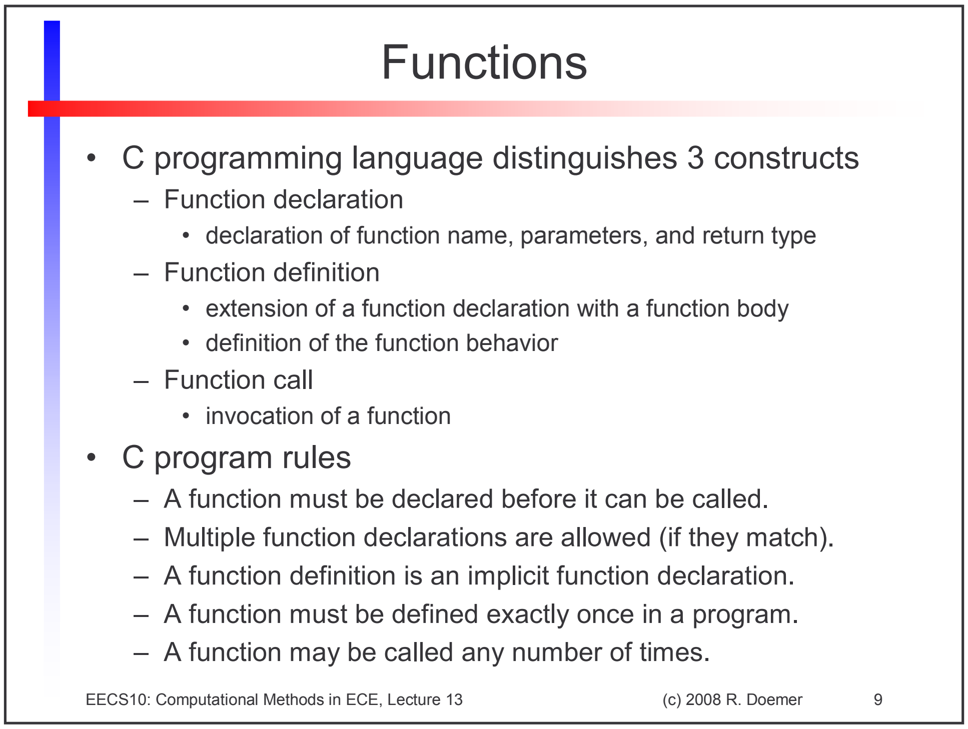
```
double Square(double p)
{
  double r;
  r = p * p;
  return r;
}
```

Functions

- Function call
  - expression invoking a function
  - supplies arguments for formal parameters
  - invokes the function
  - result is the value returned by the function
- Example:

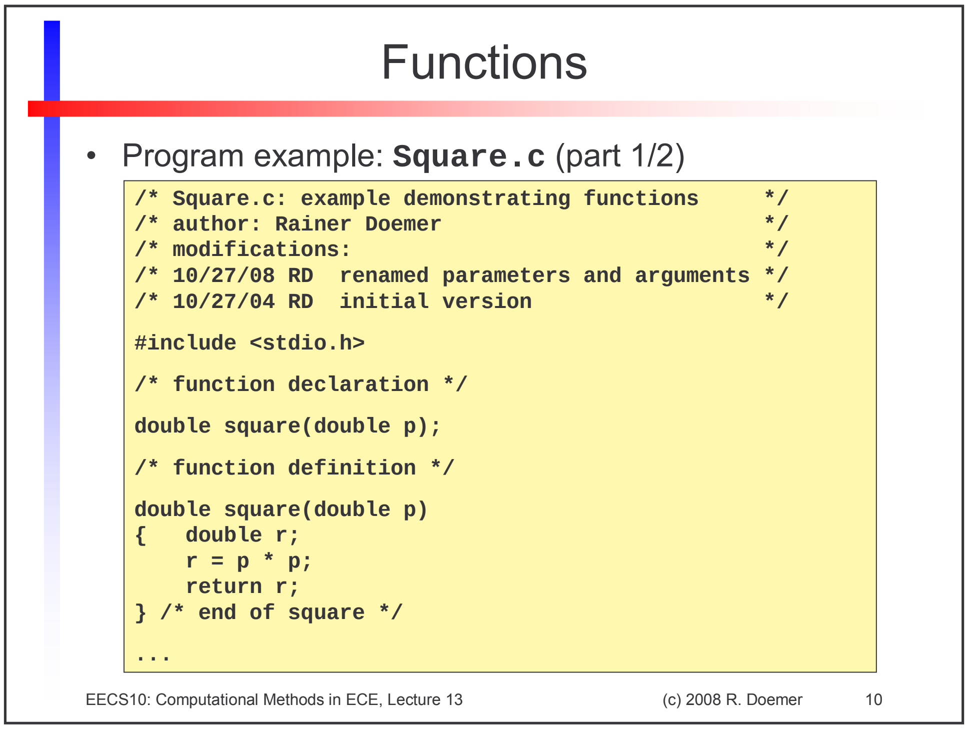
```
double a, b;
b = Square(a);
```

  - function **`Square`** is called
  - argument **`a`** is passed for parameter **`p`** (by value)
  - value returned by the function is assigned to **`b`**



Functions

- C programming language distinguishes 3 constructs
  - Function declaration
    - declaration of function name, parameters, and return type
  - Function definition
    - extension of a function declaration with a function body
    - definition of the function behavior
  - Function call
    - invocation of a function
- C program rules
  - A function must be declared before it can be called.
  - Multiple function declarations are allowed (if they match).
  - A function definition is an implicit function declaration.
  - A function must be defined exactly once in a program.
  - A function may be called any number of times.



Functions

- Program example: **`Square.c`** (part 1/2)

```
/* Square.c: example demonstrating functions     */
/* author: Rainer Doemer                         */
/* modifications:                                */
/* 10/27/08 RD  renamed parameters and arguments */
/* 10/27/04 RD  initial version                  */

#include <stdio.h>

/* function declaration */

double square(double p);

/* function definition */

double square(double p)
{   double r;
    r = p * p;
    return r;
} /* end of square */

...
```

Functions

- Program example: **`Square.c`** (part 2/2)

```
...
/* main function */

int main(void)
{  /* variable definitions */
    double a, b;

    /* input section */
    printf("Please enter a value for the argument: ");
    scanf("%lf", &a);

    /* computation section */
    b = square(a);

    /* output section */
    printf("The square of %g is %g.\n", a, b);

    /* exit */
    return 0;
} /* end of main */

/* EOF */
```

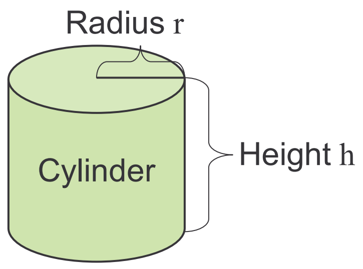
Functions

- Example session: **`Square.c`**

```
% vi Square.c
% gcc Square.c -o Square -Wall -ansi
% Square
Please enter a value for the argument: 3
The square of 3 is 9.
% Square
Please enter a value for the argument: 5.5
The square of 5.5 is 30.25.
%
```

Functions

- Hierarchy of Functions
  - functions call other functions
- Example:
  Cylinder calculations
  - given radius and height
  - calculate surface and volume



– Circle constant	$\pi = 3.14159265...$
– Circle perimeter	$f_p(r) = 2 \times \pi \times r$
– Circle area	$f_a(r) = \pi \times r^2$
– Cylinder surface	$f_s(r, h) = f_p(r) \times h + 2 \times f_a(r)$
– Cylinder volume	$f_v(r, h) = f_a(r) \times h$

Functions

- Program example: **`Cylinder.c`** (part 1/3)

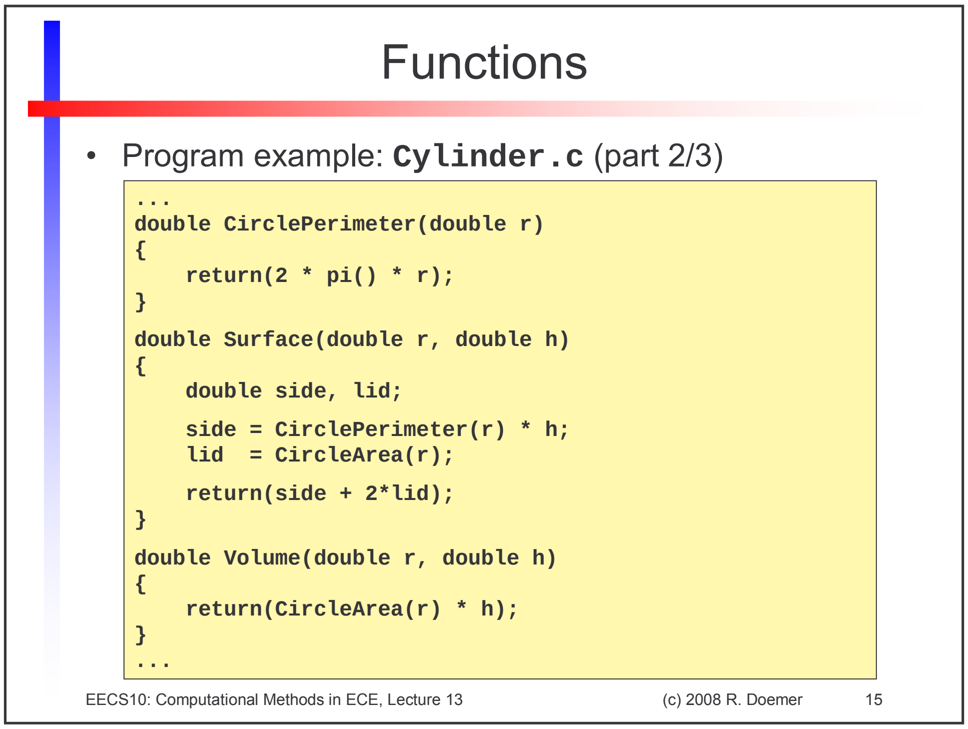
```
/* Cylinder.c: cylinder functions       */
/* author: Rainer Doemer                */
/* modifications:                       */
/* 10/25/05 RD  initial version         */

#include <stdio.h>

/* cylinder functions */

double pi(void)
{
    return(3.1415927);
}

double CircleArea(double r)
{
    return(pi() * r * r);
}
...
```

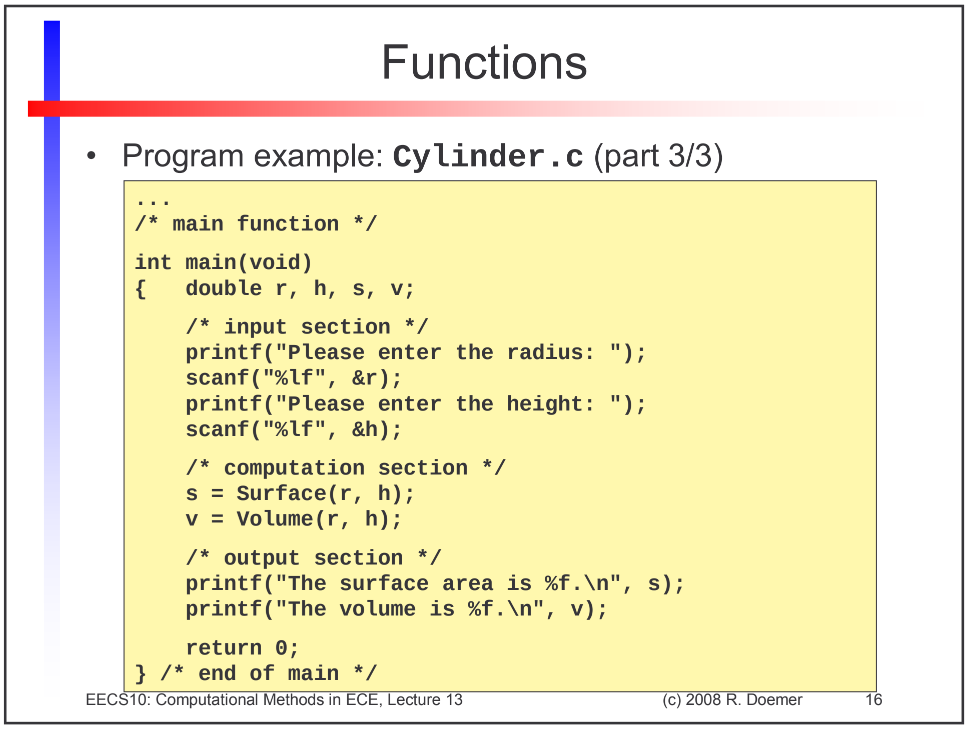


Functions

- Program example: **`Cylinder.c`** (part 2/3)

```
...
double CirclePerimeter(double r)
{
    return(2 * pi() * r);
}
double Surface(double r, double h)
{
    double side, lid;
    side = CirclePerimeter(r) * h;
    lid  = CircleArea(r);
    return(side + 2*lid);
}
double Volume(double r, double h)
{
    return(CircleArea(r) * h);
}
...
```

EECS10: Computational Methods in ECE, Lecture 13   (c) 2008 R. Doemer   15

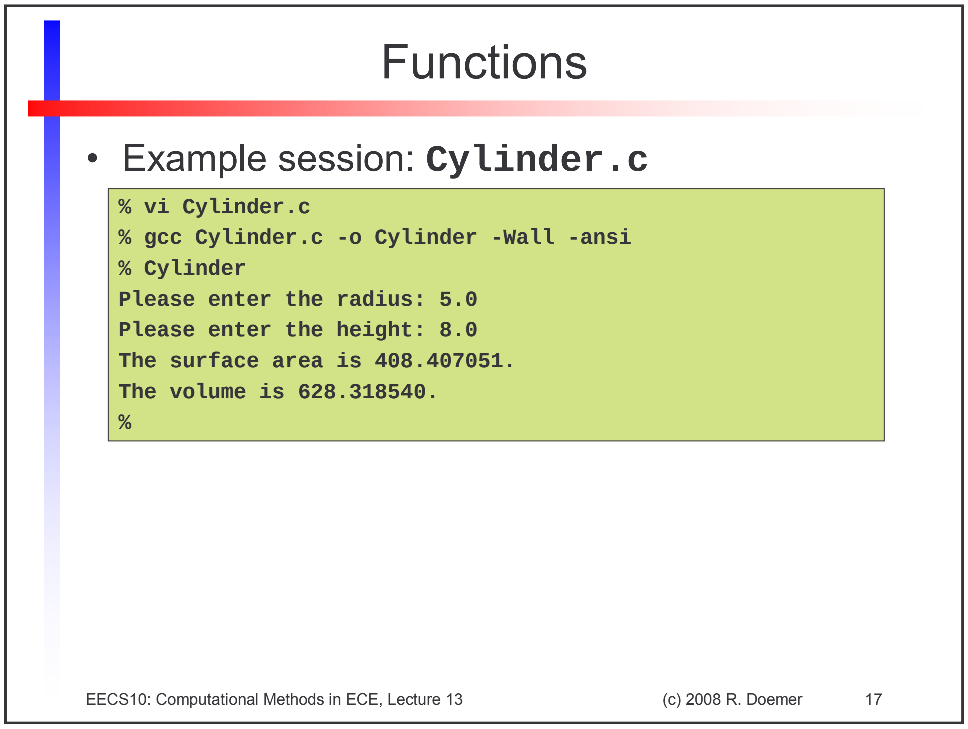


Functions

- Program example: **`Cylinder.c`** (part 3/3)

```
...
/* main function */
int main(void)
{   double r, h, s, v;
    /* input section */
    printf("Please enter the radius: ");
    scanf("%lf", &r);
    printf("Please enter the height: ");
    scanf("%lf", &h);
    /* computation section */
    s = Surface(r, h);
    v = Volume(r, h);
    /* output section */
    printf("The surface area is %f.\n", s);
    printf("The volume is %f.\n", v);
    return 0;
} /* end of main */
```

EECS10: Computational Methods in ECE, Lecture 13   (c) 2008 R. Doemer   16



Functions

- Example session: **`Cylinder.c`**

```
% vi Cylinder.c
% gcc Cylinder.c -o Cylinder -Wall -ansi
% Cylinder
Please enter the radius: 5.0
Please enter the height: 8.0
The surface area is 408.407051.
The volume is 628.318540.
%
```

EECS10: Computational Methods in ECE, Lecture 13 (c) 2008 R. Doemer 17