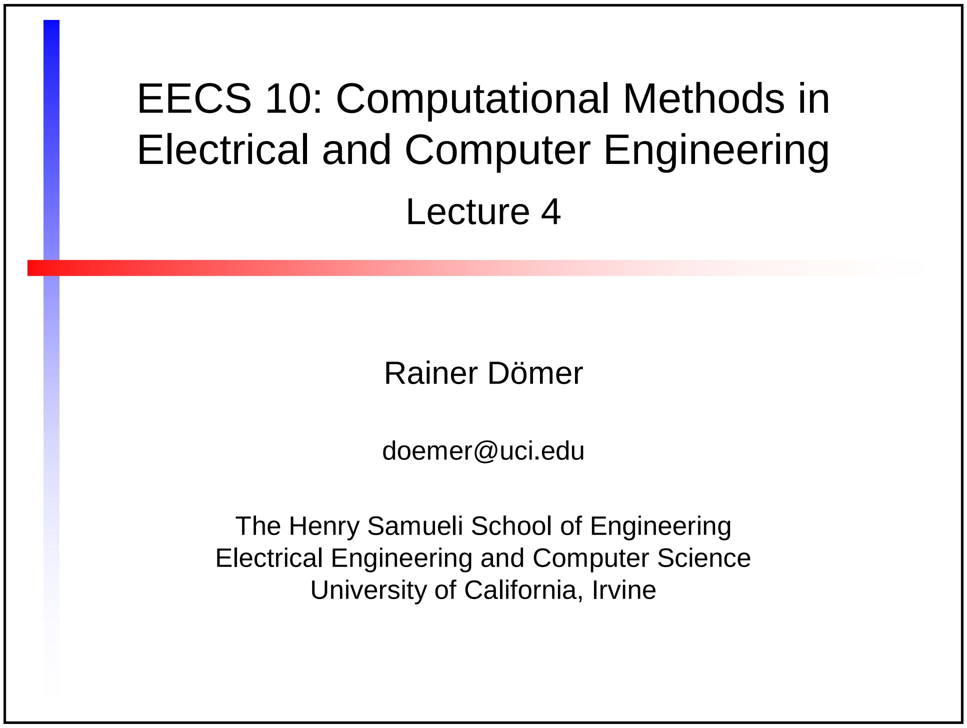


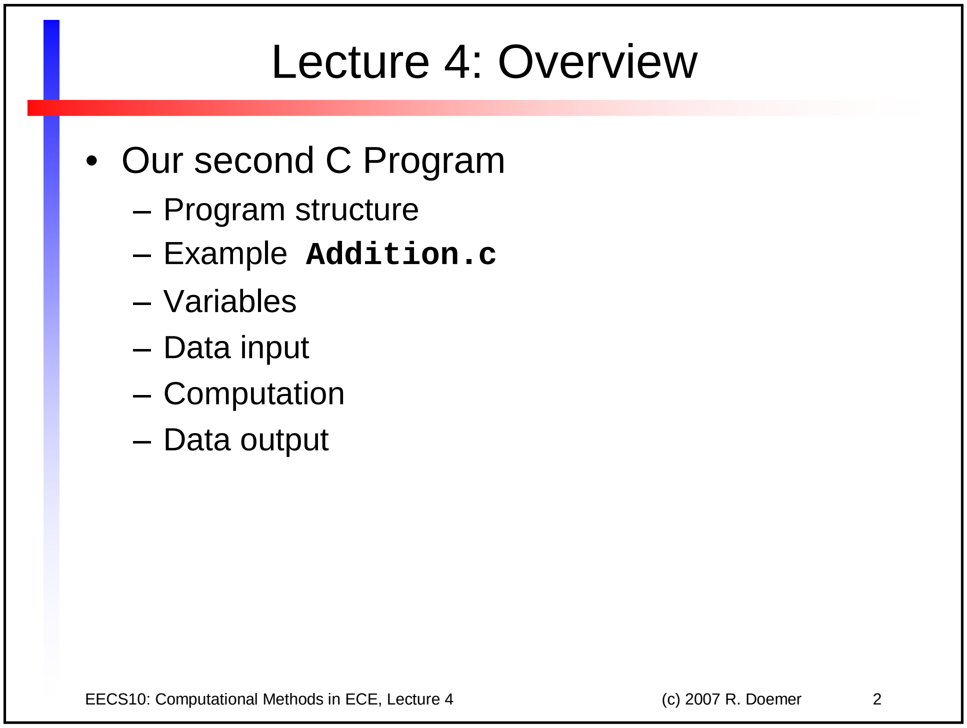
EECS 10: Computational Methods in Electrical and Computer Engineering

Lecture 4

Rainer Dömer

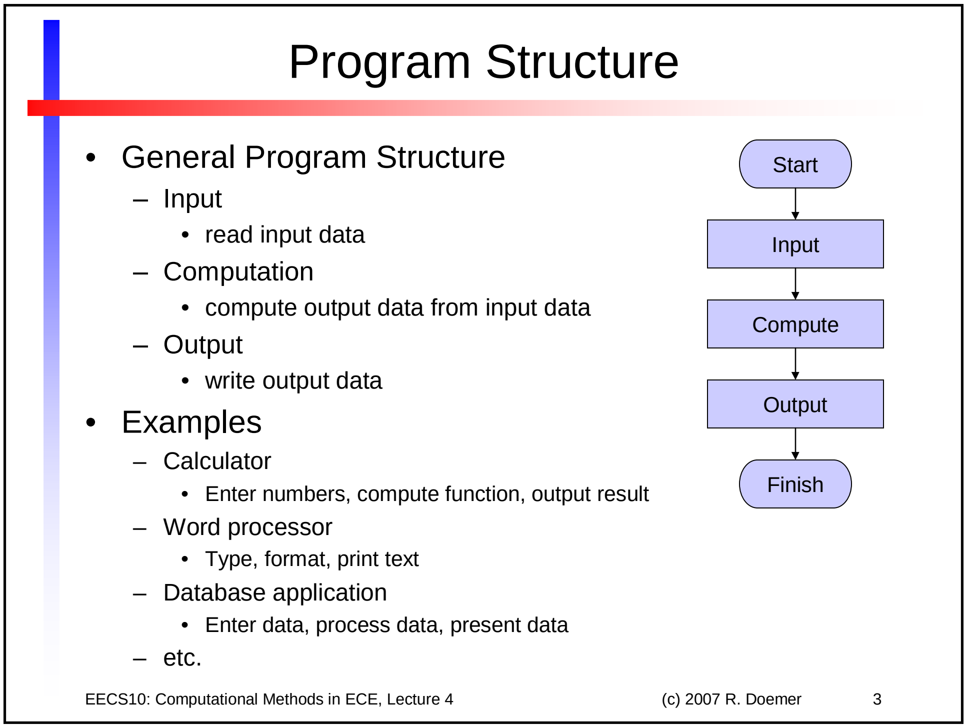
doemer@uci.edu

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



Lecture 4: Overview

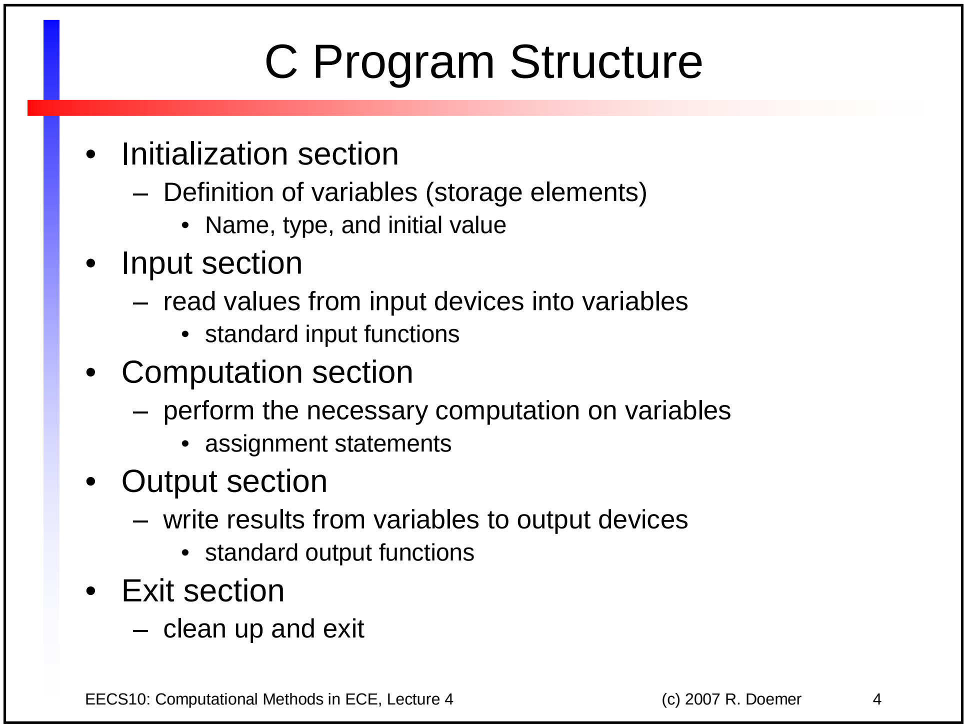
- Our second C Program
  - Program structure
  - Example **`Addition.c`**
  - Variables
  - Data input
  - Computation
  - Data output



Program Structure

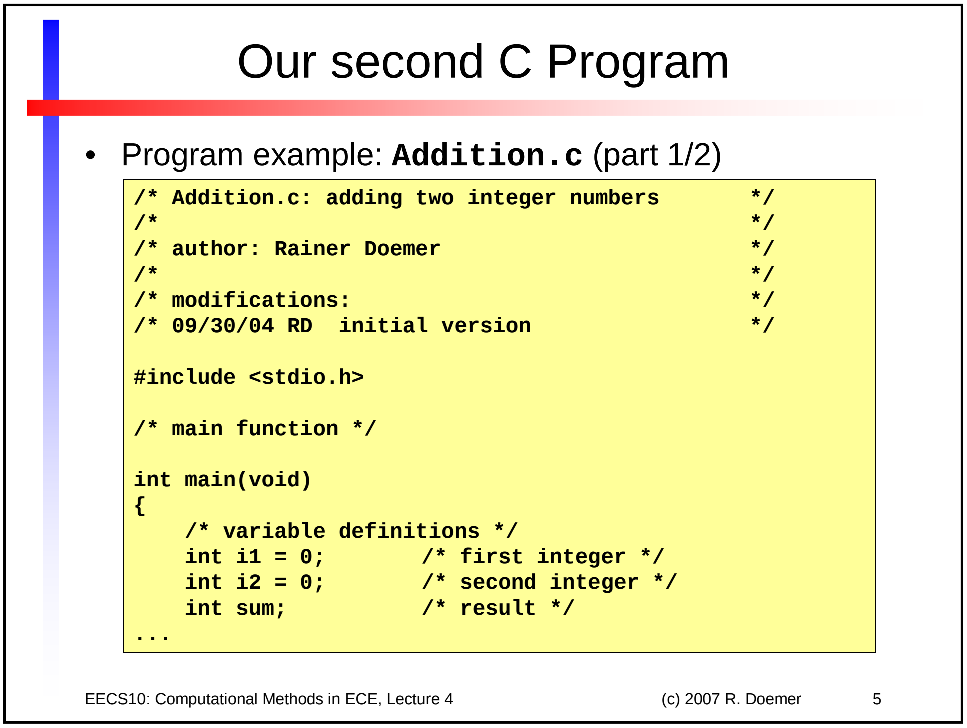
- General Program Structure
  - Input
    - read input data
  - Computation
    - compute output data from input data
  - Output
    - write output data
- Examples
  - Calculator
    - Enter numbers, compute function, output result
  - Word processor
    - Type, format, print text
  - Database application
    - Enter data, process data, present data
  - etc.

Start → Input → Compute → Output → Finish



C Program Structure

- Initialization section
  - Definition of variables (storage elements)
    - Name, type, and initial value
- Input section
  - read values from input devices into variables
    - standard input functions
- Computation section
  - perform the necessary computation on variables
    - assignment statements
- Output section
  - write results from variables to output devices
    - standard output functions
- Exit section
  - clean up and exit



Our second C Program

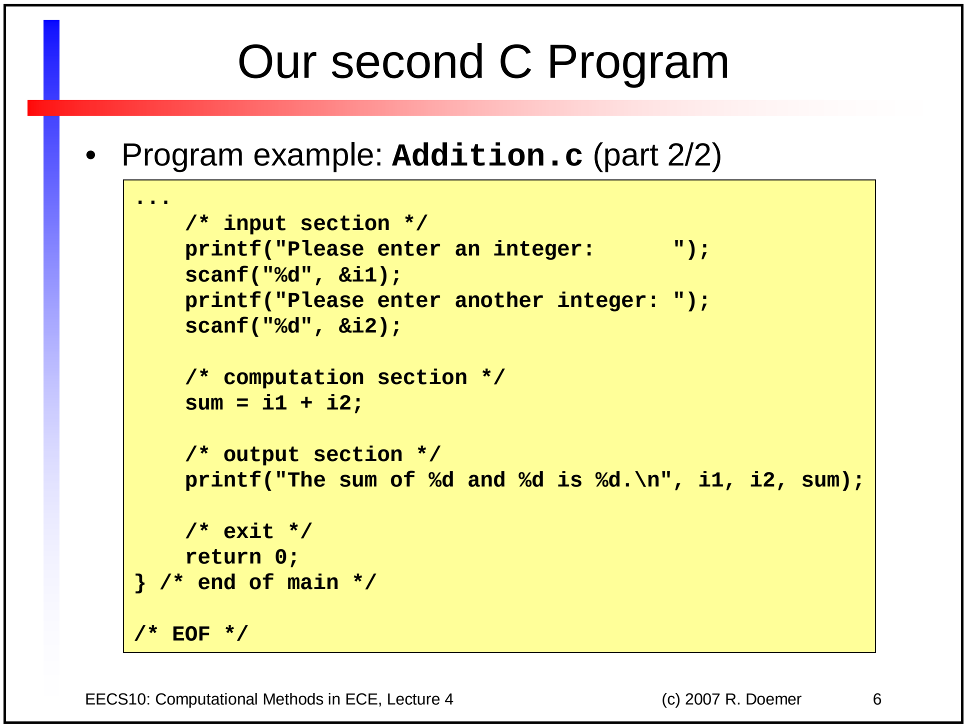
- Program example: **`Addition.c`** (part 1/2)

```
/* Addition.c: adding two integer numbers       */
/*                                              */
/* author: Rainer Doemer                        */
/*                                              */
/* modifications:                               */
/* 09/30/04 RD  initial version                 */

#include <stdio.h>

/* main function */

int main(void)
{
    /* variable definitions */
    int i1 = 0;           /* first integer */
    int i2 = 0;           /* second integer */
    int sum;              /* result */
...
```



Our second C Program

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

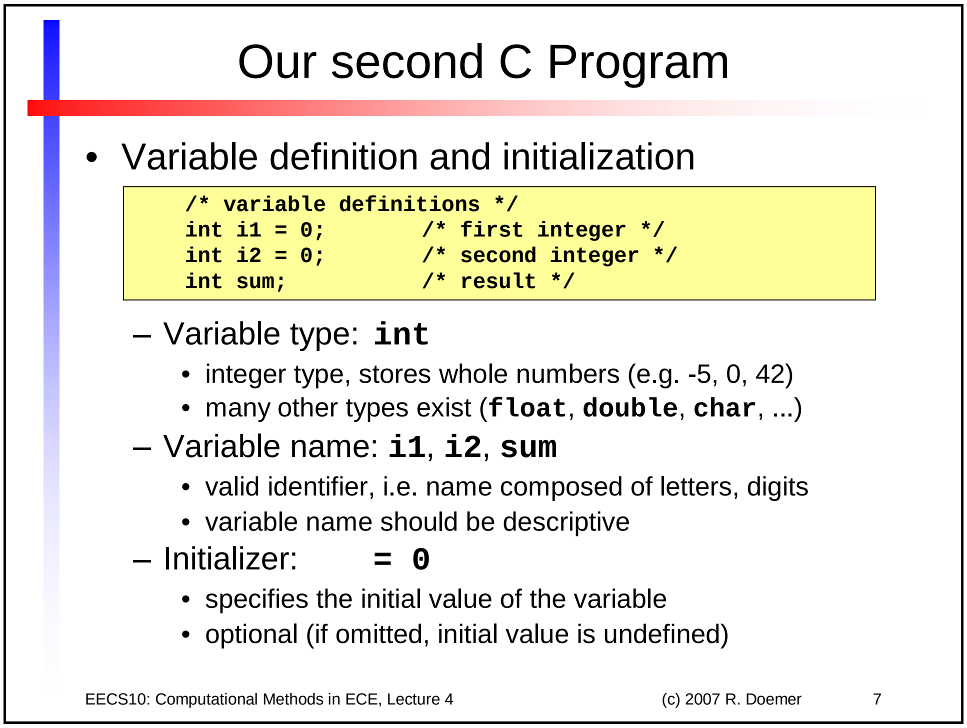
```
...
    /* input section */
    printf("Please enter an integer:      ");
    scanf("%d", &i1);
    printf("Please enter another integer: ");
    scanf("%d", &i2);

    /* computation section */
    sum = i1 + i2;

    /* output section */
    printf("The sum of %d and %d is %d.\n", i1, i2, sum);

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

/* EOF */
```

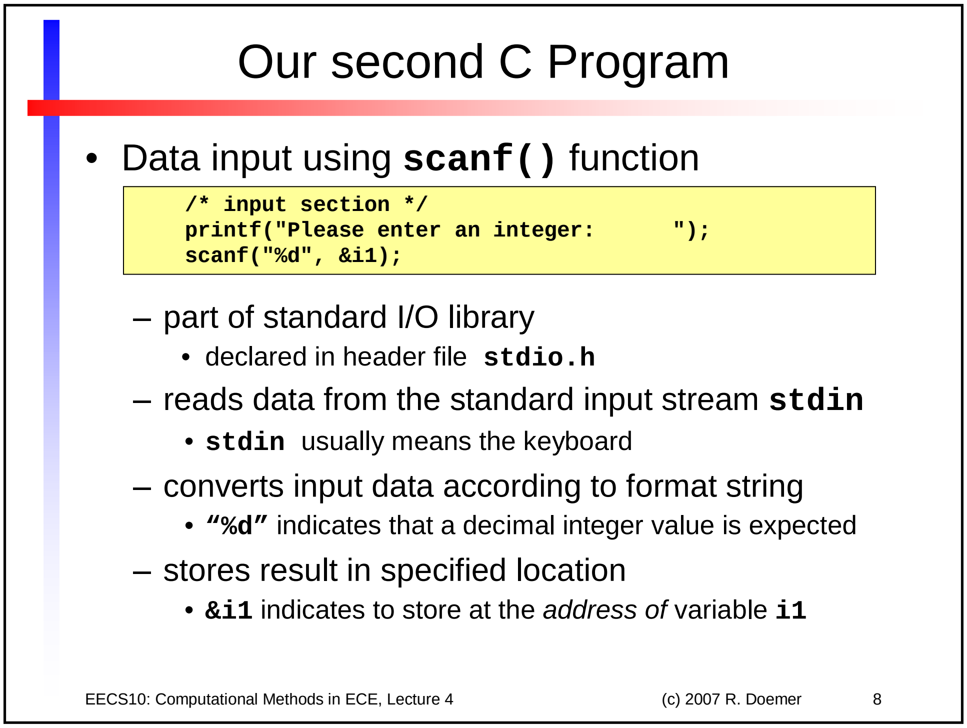


Our second C Program

- Variable definition and initialization

```
/* variable definitions */
int i1 = 0;       /* first integer */
int i2 = 0;       /* second integer */
int sum;          /* result */
```

  - Variable type: **`int`**
    - integer type, stores whole numbers (e.g. -5, 0, 42)
    - many other types exist (**`float`**, **`double`**, **`char`**, ...)
  - Variable name: **`i1`**, **`i2`**, **`sum`**
    - valid identifier, i.e. name composed of letters, digits
    - variable name should be descriptive
  - Initializer: **`= 0`**
    - specifies the initial value of the variable
    - optional (if omitted, initial value is undefined)

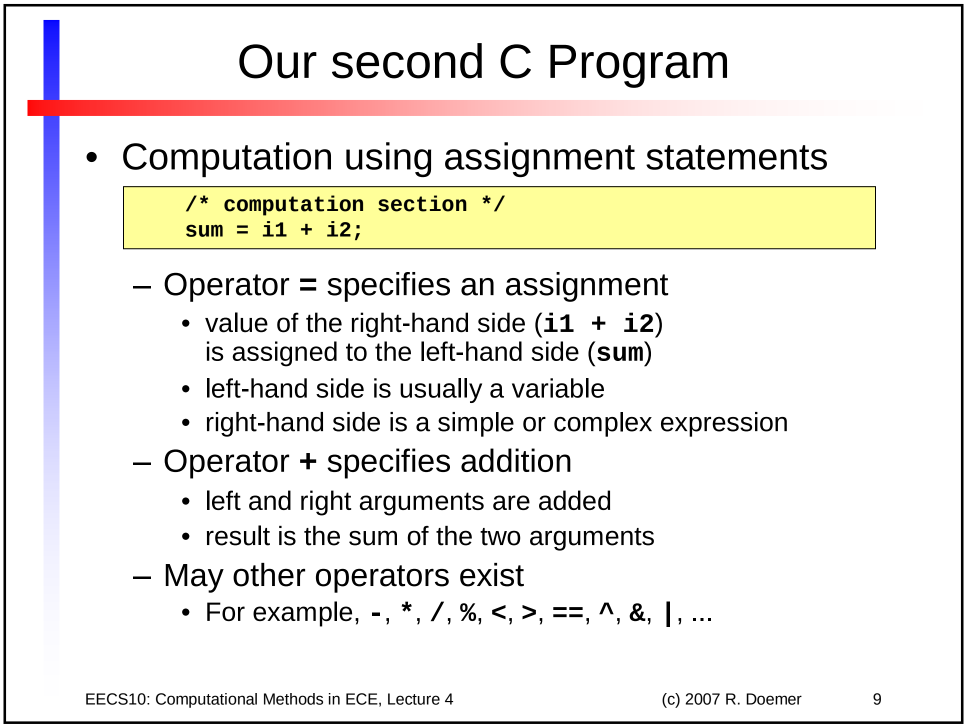


Our second C Program

- Data input using **`scanf()`** function

```
/* input section */
printf("Please enter an integer:      ");
scanf("%d", &i1);
```

  - part of standard I/O library
    - declared in header file **`stdio.h`**
  - reads data from the standard input stream **`stdin`**
    - **`stdin`** usually means the keyboard
  - converts input data according to format string
    - **`“%d”`** indicates that a decimal integer value is expected
  - stores result in specified location
    - **`&i1`** indicates to store at the *address of* variable **`i1`**



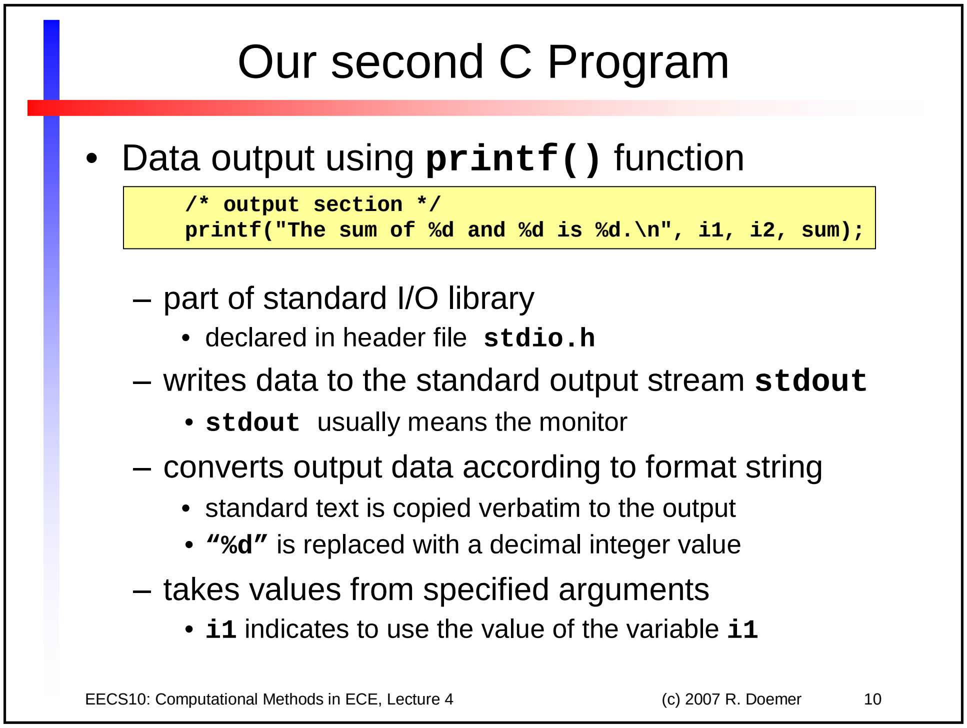
Our second C Program

- Computation using assignment statements

```
/* computation section */
sum = i1 + i2;
```

  - Operator **=** specifies an assignment
    - value of the right-hand side (**`i1 + i2`**) is assigned to the left-hand side (**`sum`**)
    - left-hand side is usually a variable
    - right-hand side is a simple or complex expression
  - Operator **+** specifies addition
    - left and right arguments are added
    - result is the sum of the two arguments
  - May other operators exist
    - For example, **`-`**, **`*`**, **`/`**, **`%`**, **`<`**, **`>`**, **`==`**, **`^`**, **`&`**, **`|`**, ...

EECS10: Computational Methods in ECE, Lecture 4 (c) 2007 R. Doemer 9



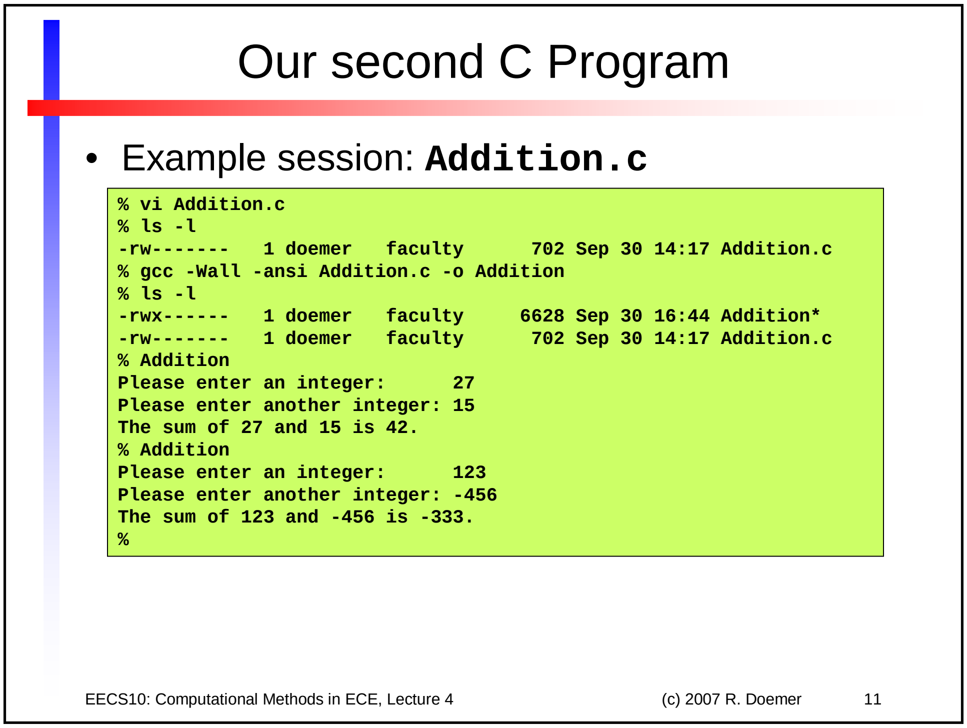
Our second C Program

- Data output using **`printf()`** function

```
/* output section */
printf("The sum of %d and %d is %d.\n", i1, i2, sum);
```

  - part of standard I/O library
    - declared in header file **`stdio.h`**
  - writes data to the standard output stream **`stdout`**
    - **`stdout`** usually means the monitor
  - converts output data according to format string
    - standard text is copied verbatim to the output
    - **“%d”** is replaced with a decimal integer value
  - takes values from specified arguments
    - **`i1`** indicates to use the value of the variable **`i1`**

EECS10: Computational Methods in ECE, Lecture 4 (c) 2007 R. Doemer 10



Our second C Program

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

```
% vi Addition.c
% ls -l
-rw-------   1 doemer   faculty      702 Sep 30 14:17 Addition.c
% gcc -Wall -ansi Addition.c -o Addition
% ls -l
-rwx------   1 doemer   faculty     6628 Sep 30 16:44 Addition*
-rw-------   1 doemer   faculty      702 Sep 30 14:17 Addition.c
% Addition
Please enter an integer:      27
Please enter another integer: 15
The sum of 27 and 15 is 42.
% Addition
Please enter an integer:      123
Please enter another integer: -456
The sum of 123 and -456 is -333.
%
```