

# EECS 10: Computational Methods in Electrical and Computer Engineering

## Lecture 7

Rainer Dömer

doemer@uci.edu

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

## Lecture 7: Overview

- Formatted Input and Output
- Programming Principles
  - Algorithm and control flow
- Structured Programming
  - Control flow chart
  - Sequential execution
  - Conditional execution
    - `if` statement
    - `if-else` statement
    - `switch` statement
  - Structured Program Composition
  - Examples `Grade.c`, `Grade2.c`

## Formatted Input and Output

- Formatted Input
  - Format specifiers for `scanf()`
  - Detailed formatting of integral values
  - Detailed formatting of floating-point values
- Formatted Output
  - Format specifiers for `printf()`
  - Detailed formatting of integral values
  - Detailed formatting of floating-point values
- Example `Formatting.c`

EECS10: Computational Methods in ECE, Lecture 7

(c) 2019 R. Doemer

3

## Formatted Input

- Formatted input using `scanf()`
  - standard format specifier for integral values
    - (unsigned) long long    `%llu`    `%lld`
    - (unsigned) long        `%lu`     `%ld`
    - (unsigned) int         `%u`      `%d`
    - (unsigned) short       `%hu`     `%hd`
    - (unsigned) char        `%c` (reads a character)
  - standard format specifier for floating point values
    - long double            `%Lf`
    - double                 `%lf`
    - float                  `%f`

EECS10: Computational Methods in ECE, Lecture 7

(c) 2019 R. Doemer

4

## Formatted Output

- Formatted output using `printf()`
  - standard format specifier for integral values
    - (unsigned) long long    `%llu`    `%lld`
    - (unsigned) long        `%lu`     `%ld`
    - (unsigned) int         `%u`      `%d`
    - (unsigned) short       `%hu`     `%hd`
    - (unsigned) char        `%c` (prints a character)
  - standard format specifier for floating point values
    - long double            `%Lf`
    - double                 `%f`
    - float                  `%f`

EECS10: Computational Methods in ECE, Lecture 7

(c) 2019 R. Doemer

5

## Formatted Output

- *Detailed* formatting sequence for integral values
  - *% flags width length conversion*
  - *flags*
    - (none)    standard formatting (right-justified)
    - -         left-justified output
    - +         leading plus-sign for positive values
    - 0         leading zeros
  - field *width*
    - (none)    minimum number of characters needed
    - integer    width of field to be filled with output
  - *length* modifier
    - (none)    `int` type
    - `h`        `short int` type
    - `l`        `long int` type
    - `ll`       `long long int` type
  - *conversion* specifier
    - `d`        signed decimal value
    - `u`        unsigned decimal value
    - `o`        (unsigned) octal value
    - `x`        (unsigned) hexadecimal value using characters `0-9, a-f`
    - `X`        (unsigned) hexadecimal value using characters `0-9, A-F`

EECS10: Computational Methods in ECE, Lecture 7

(c) 2019 R. Doemer

6

## Formatted Output

- *Detailed* formatting sequence for floating-point values
  - *% flags width precision length conversion*
  - **flags**
    - (none) standard formatting (right-justified)
    - - left-justified output
    - + leading plus-sign for positive values
    - 0 leading zeros
  - **field width**
    - (none) minimum number of characters needed
    - integer width of field to be filled with output
  - **precision**
    - (none) default precision (e.g. 6)
    - .int number of digits after decimal point (for **f**, **e**, or **E**), maximum number of significant digits (for **g**, or **G**)
  - **length** modifier
    - (none) **float** or **double** type
    - **L** **long double** type
  - **conversion** specifier
    - **f** standard floating-point notation (fixed-point)
    - **e** or **E** exponential notation (using **e** or **E**)
    - **g** or **G** standard or exponential notation (using **e** or **E**)

EECS10: Computational Methods in ECE, Lecture 7

(c) 2019 R. Doemer

7

## Formatted Output

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

```

/* Formatting.c: formatted output demo          */
/* author: Rainer Doemer                       */
/* modifications:                              */
/* 10/19/04 RD initial version                 */

#include <stdio.h>

/* main function */

int main(void)
{
    /* output section */
    printf("42 formatted as |%d|:   |%d|\n", 42);
    printf("42 formatted as |%8d|:  |%8d|\n", 42);
    printf("42 formatted as |%-8d|: |%-8d|\n", 42);
    printf("42 formatted as |%+8d|: |%+8d|\n", 42);
    printf("42 formatted as |%08d|: |%08d|\n", 42);
    printf("42 formatted as |%x|:   |%x|\n", 42);
    printf("42 formatted as |%o|:   |%o|\n", 42);
    ...
}

```

EECS10: Computational Methods in ECE, Lecture 7

(c) 2019 R. Doemer

8

## Formatted Output

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

```

...
printf("\n");
printf("123.456 formatted as |%f|:      |%f|\n", 123.456);
printf("123.456 formatted as |%e|:      |%e|\n", 123.456);
printf("123.456 formatted as |%g|:      |%g|\n", 123.456);
printf("123.456 formatted as |%.12.4f|: |%.12.4f|\n",
      123.456);
printf("123.456 formatted as |%.12.4e|: |%.12.4e|\n",
      123.456);
printf("123.456 formatted as |%.12.4g|: |%.12.4g|\n",
      123.456);

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

/* EOF */

```

## Formatted Output

- Example session: `Formatting.c`

```

% vi Formatting.c
% gcc Formatting.c -o Formatting -Wall -ansi
% ./Formatting
42 formatted as |%d|:      |42|
42 formatted as |%8d|:      |      42|
42 formatted as |%-8d|:     |42      |
42 formatted as |%+8d|:     |      +42|
42 formatted as |%08d|:     |00000042|
42 formatted as |%x|:      |2a|
42 formatted as |%o|:      |52|

123.456 formatted as |%f|:      |123.456000|
123.456 formatted as |%e|:      |1.234560e+02|
123.456 formatted as |%g|:      |123.456|
123.456 formatted as |%.12.4f|: |      123.4560|
123.456 formatted as |%.12.4e|: |  1.2346e+02|
123.456 formatted as |%.12.4g|: |      123.5|
%

```

## Programming Principles

- Thorough *understanding* of the problem
- *Problem definition*
  - Input data
  - Output data
- *Algorithm*: Procedure to solve the problem
  - Detailed set of *actions* to perform
  - Specification of *order* in which to perform the actions
  - Termination after a *finite* number of steps
- *Pseudo code*: Planning a program
  - Informal (English) description of steps in an algorithm
  - Example: Cake baking recipe
- *Control flow*
  - Detailed execution order of steps in the program
- *Program*: Instructions for the computer
  - Formal description in programming language
    - Statements (steps, actions)
    - Control structures (flow of control)

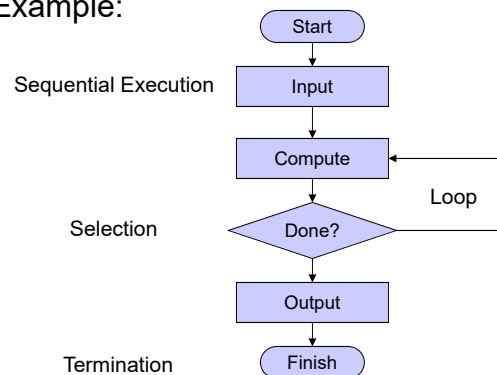
EECS10: Computational Methods in ECE, Lecture 7

(c) 2019 R. Doemer

11

## Structured Programming

- Control Flow Chart
  - Graphical representation of program control flow
  - Example:



EECS10: Computational Methods in ECE, Lecture 7

(c) 2019 R. Doemer

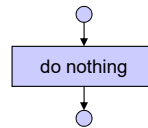
12

## Structured Programming

- Empty statement blocks
  - empty compound statement
  - does nothing (no operation, no-op)
  - Example:

Flow chart:

```
{
  /* nothing */
}
```



EECS10: Computational Methods in ECE, Lecture 7

(c) 2019 R. Doemer

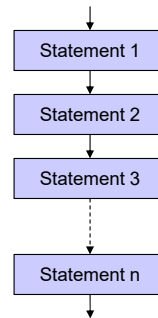
13

## Structured Programming

- Sequential execution in C
  - Statement blocks: *Compound statements*
  - Sequence of statements grouped by braces: { }
- Example:

Flow chart:

```
{
  /* statement 1 */
  /* statement 2 */
  /* statement 3 */
  /* ... */
  /* statement n */
}
```



EECS10: Computational Methods in ECE, Lecture 7

(c) 2019 R. Doemer

14

## Structured Programming

- Sequential execution in C
  - Statement blocks: *Compound statements*
  - Sequence of statements grouped by braces: { }
- *Indentation* increases readability of the code
  - proper indentation is highly recommended!
- Example:

```

/* some statements... */
if (x < 0) {
    printf("%d is negative!", x);
    /* handle negative values of x... */
    if (x < -100) {
        printf("%d is too small!", x);
        /* handle the problem... */
    } /* fi */
} /* fi */
if (x > 0) {
    printf("%d is positive!", x);
    /* handle positive values of x... */
} /* fi */
/* more statements... */

```

EECS10: Computational Methods in ECE, Lecture 7

(c) 2019 R. Doemer

15

## Structured Programming

- Sequential execution in C
  - Statement blocks: *Compound statements*
  - Sequence of statements grouped by braces: { }
- *Indentation* increases readability of the code
  - proper indentation is highly recommended!
- Example:

```

/* some statements... */
if (x < 0) {
    printf("%d is negative!", x);
    /* handle negative values of x... */
    if (x < -100) {
        printf("%d is too small!", x);
        /* handle the problem... */
    } /* fi */
} /* fi */
if (x > 0) {
    printf("%d is positive!", x);
    /* handle positive values of x... */
} /* fi */
/* more statements... */

```

indentation level 0

indentation level 1 →

indentation level 2 → →

indentation level 1 →

indentation level 0

indentation level 1 →

indentation level 0

EECS10: Computational Methods in ECE, Lecture 7

(c) 2019 R. Doemer

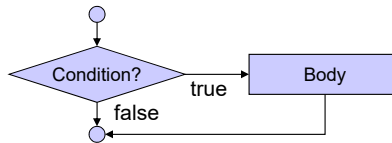
16



## Structured Programming

- Selection: **if** statement

– Flow chart:



– Example:

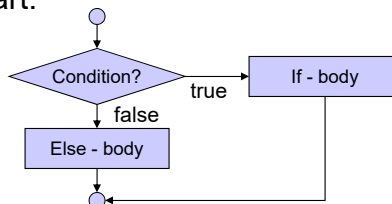
```

if (grade >= 60)
{ printf("You passed.");
} /* fi */
  
```

## Structured Programming

- Selection: **if-else** statement

– Flow chart:



– Example:

```

if (grade >= 60)
{ printf("You passed.");
} /* fi */
else
{ printf("You failed.");
} /* esle */
  
```

## Structured Programming

- Selection: **switch** statement
  - Flow chart:
  - Example:
 

```
switch(LetterGrade)
{ case 'A':
  { printf("Excellent!");
    break; }
  case 'B':
  case 'C':
  case 'D':
    { printf("Passed.");
      break; }
  case 'F':
    { printf("Failed!");
      break; }
  default:
    { printf("Invalid grade!");
      break; }
} /* hctiws */
```

EECS10: Computational Methods in ECE, Lecture 7 (c) 2019 R. Doemer 19

## Structured Program Composition

- Initial flow chart
  - Start
  - Program body
  - Finish
- Statement sequences
  - Statement blocks can be concatenated
  - Sequential execution
- Nested control structures
  - control structures can be placed wherever statement blocks can be placed in the code

EECS10: Computational Methods in ECE, Lecture 7 (c) 2019 R. Doemer 20

## Structured Program Composition

- Example:
  - Initial flow chart

```

graph TD
    Start([Start]) --> Process[Process]
    Process --> End([End])
    
```

EECS10: Computational Methods in ECE, Lecture 7 (c) 2019 R. Doemer 21

## Structured Program Composition

- Example:
  - Sequential composition

```

graph TD
    Start([Start]) --> P1[Process 1]
    P1 --> P2[Process 2]
    subgraph Box [ ]
        P1
        P2
    end
    Box --> End([End])
    
```

EECS10: Computational Methods in ECE, Lecture 7 (c) 2019 R. Doemer 22

## Structured Program Composition

- Example:
  - insertion of another sequential statement

The flowchart illustrates a sequential process. It starts with an oval representing the beginning of the program. An arrow points down to a rectangular box representing a statement. This box is enclosed in a dashed-line rectangle, indicating it is the focus of the example. Below this box is another rectangular box, representing a second sequential statement. An arrow points from the second box to a third rectangular box, representing a third sequential statement. Finally, an arrow points from the third box to an oval representing the end of the program.

EECS10: Computational Methods in ECE, Lecture 7 (c) 2019 R. Doemer 23

## Structured Program Composition

- Example:
  - insertion of **if - else** statement

The flowchart illustrates a program structure with an if-else statement. It starts with an oval representing the beginning of the program. An arrow points down to a rectangular box representing a statement. Below this box is a diamond-shaped decision node. This diamond node is enclosed in a dashed-line rectangle, indicating it is the focus of the example. An arrow points from the diamond to a rectangular box on the right, representing the 'else' branch. Another arrow points from the diamond to a rectangular box below it, representing the 'if' branch. An arrow from the 'else' branch box loops back to the entry point of the dashed rectangle, just above the 'if' branch box. Below the 'if' branch box is a third rectangular box, representing a statement following the if-else construct. Finally, an arrow points from the third box to an oval representing the end of the program.

EECS10: Computational Methods in ECE, Lecture 7 (c) 2019 R. Doemer 24

## Structured Program Composition

- Example:
  - insertion of sequential statement

The flowchart shows a sequence of operations: a start node (oval), a process node (rectangle), a decision node (diamond), another process node, and a final node (oval). A loop is formed by a process node and a decision node. A dashed box highlights the insertion of a new process node into the loop. The flow starts at the top oval, goes to the first process node, then to the decision diamond. From the diamond, it goes to the process node in the dashed box, then to the second process node, then to the second decision diamond, and finally back to the first process node. The flow then proceeds to the final oval.

EECS10: Computational Methods in ECE, Lecture 7 (c) 2019 R. Doemer 25

## Structured Program Composition

- Example:
  - insertion of **if - else** statement

The flowchart shows a sequence of operations: a start node (oval), a process node, a decision node, another process node, and a final node (oval). A loop is formed by a process node and a decision node. A dashed box highlights the insertion of an if-else statement into the loop. The flow starts at the top oval, goes to the first process node, then to the decision diamond. From the diamond, it goes to the process node in the dashed box, then to the second decision diamond, then to the process node in the dashed box, then to the second decision diamond, and finally back to the first process node. The flow then proceeds to the final oval.

EECS10: Computational Methods in ECE, Lecture 7 (c) 2019 R. Doemer 26

## Structured Program Composition

- Example:
  - insertion of sequential statement

The flowchart shows a sequence of operations: a start node, a process box, a decision diamond, a process box, a decision diamond, a process box, a loop body (two process boxes), a process box, and an end node. A dashed box highlights the loop body. An arrow from the top decision diamond points to the first process box of the loop body. An arrow from the bottom decision diamond points to the process box immediately following the loop body.

EECS10: Computational Methods in ECE, Lecture 7 (c) 2019 R. Doemer 27

## Structured Program Composition

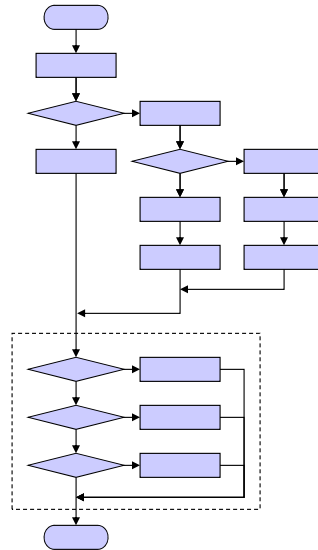
- Example:
  - insertion of sequential statement (twice)

The flowchart shows a sequence of operations: a start node, a process box, a decision diamond, a process box, a decision diamond, a loop body (three process boxes), a process box, and an end node. A dashed box highlights the loop body. An arrow from the top decision diamond points to the first process box of the loop body. An arrow from the bottom decision diamond points to the process box immediately following the loop body.

EECS10: Computational Methods in ECE, Lecture 7 (c) 2019 R. Doemer 28

## Structured Program Composition

- Example:
  - insertion of **switch** statement
  - etc. ...



EECS10: Computational Methods in ECE, Lecture 7

(c) 2019 R. Doemer

29

## Example Program

- Grade calculation: **Grade.c** (part 1/3)

```

/* Grade.c: convert score into letter grade */
/* author: Rainer Doemer */
/* modifications: */
/* 10/17/04 RD initial version */

#include <stdio.h>

/* main function */
int main(void)
{
    /* variable definitions */
    int score = 0;
    char grade;

    /* input section */
    while (score < 1 || score > 100)
    { printf("Please enter your score (1-100): ");
      scanf("%d", &score);
    } /* elihw */

    ...
  
```

EECS10: Computational Methods in ECE, Lecture 7

(c) 2019 R. Doemer

30

## Example Program

- Grade calculation: `Grade.c` (part 2/3)

```
...
/* computation section */
if (score >= 90)
    { grade = 'A'; }
else
    { if (score >= 80)
      { grade = 'B'; }
      else
        { if (score >= 70)
          { grade = 'C'; }
          else
            { if (score >= 60)
              { grade = 'D'; }
              else
                { grade = 'F'; }
            } /* esle */
          } /* esle */
        } /* esle */
...

```

EECS10: Computational Methods in ECE, Lecture 7

(c) 2019 R. Doemer

31

## Example Program

- Grade calculation: `Grade.c` (part 3/3)

```
...
/* output section */
printf("Your letter grade is %c.\n", grade);

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

/* EOF */

```

EECS10: Computational Methods in ECE, Lecture 7

(c) 2019 R. Doemer

32



## Example Program

- Example session: `Grade.c`

```
% vi Grade.c
% gcc Grade.c -o Grade -Wall -ansi
% ./Grade
Please enter your score (1-100): 111
Please enter your score (1-100): 99
Your letter grade is A.
% Grade
Please enter your score (1-100): 85
Your letter grade is B.
% Grade
Please enter your score (1-100): 71
Your letter grade is C.
% Grade
Please enter your score (1-100): 69
Your letter grade is D.
% Grade
Please enter your score (1-100): 55
Your letter grade is F.
%
```

## Example Program

- Grade calculation: `Grade2.c` (part 1/3)

```
/* Grade2.c: convert score into letter grade */
/* author: Rainer Doemer */
/* modifications: */
/* 10/18/04 RD use 'switch' statement */
/* 10/17/04 RD initial version */

#include <stdio.h>

/* main function */

int main(void)
{
    /* variable definitions */
    int score = 0;
    char grade;

    /* input section */
    while (score < 1 || score > 100)
    { printf("Please enter your score (1-100): ");
      scanf("%d", &score);
    } /* elihw */
    ...
}
```

## Example Program

- Grade calculation: `Grade2.c` (part 2/3)

```
.../* computation section */
switch (score / 10)
{ case 10:
  case 9:
    { grade = 'A';
      break; }
  case 8:
    { grade = 'B';
      break; }
  case 7:
    { grade = 'C';
      break; }
  case 6:
    { grade = 'D';
      break; }
  default:
    { grade = 'F';
      break; }
} /* hctiws */
```

EECS ...

## Example Program

- Grade calculation: `Grade2.c` (part 3/3)

```
...

/* output section */
printf("Your letter grade is %c.\n", grade);

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

/* EOF */
```

## Example Program

- Example session: `Grade2.c`

```
% cp Grade.c Grade2.c
% vi Grade2.c
% gcc Grade2.c -o Grade2 -Wall -ansi
% ./Grade2
Please enter your score (1-100): 111
Please enter your score (1-100): 99
Your letter grade is A.
% Grade2
Please enter your score (1-100): 85
Your letter grade is B.
% Grade2
Please enter your score (1-100): 71
Your letter grade is C.
% Grade2
Please enter your score (1-100): 69
Your letter grade is D.
% Grade2
Please enter your score (1-100): 55
Your letter grade is F.
%
```