

## EECS 10 Midterm Examination 1

- Please...
  - ... seat yourself with space around you
  - ... have a valid photo ID ready for inspection
- Rules
  - 50 minutes time
  - No books, no notes, no devices
  - No questions
  - If in doubt, state all assumptions clearly
- Good Luck!

EECS10: Computational Methods in ECE, Lecture 4

(c) 2017 R. Doemer

1

## 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 4.1: Overview

- 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`

## 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`

## 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 4

(c) 2017 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 4

(c) 2017 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 4

(c) 2017 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 4

(c) 2017 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 |%.12f|: |%.12f|\n",
      123.456);
printf("123.456 formatted as |%.4e|: |%.4e|\n",
      123.456);
printf("123.456 formatted as |%.4g|: |%.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 |%.12f|: |      123.4560|
123.456 formatted as |%.4e|: | 1.2346e+02|
123.456 formatted as |%.4g|: |      123.5|
%

```

## Lecture 4.2: Overview

- 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`

## 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)

## Structured Programming

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

```

graph TD
    Start([Start]) --> Input[Input]
    Input --> Compute[Compute]
    Compute --> Done{Done?}
    Done -- Loop --> Compute
    Done --> Output[Output]
    Output --> Finish([Finish])
            
```

The flowchart illustrates a control flow. It begins with a 'Start' terminal, followed by 'Input', 'Compute', and 'Output' process boxes. A decision diamond labeled 'Done?' follows. A 'Loop' path branches from the 'Done?' diamond back to the 'Compute' box. The flow ends at a 'Finish' terminal.

EECS10: Computational Methods in ECE, Lecture 4 (c) 2017 R. Doemer 13

## Structured Programming

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

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

Flow chart:

```

graph TD
    Node1(( )) --> DoNothing[do nothing]
    DoNothing --> Node2(( ))
            
```

EECS10: Computational Methods in ECE, Lecture 4 (c) 2017 R. Doemer 14

## Structured Programming

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

```

{
  /* statement 1 */

  /* statement 2 */

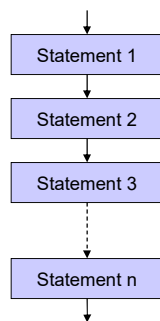
  /* statement 3 */

  /* ... */

  /* statement n */
}

```

Flow chart:



EECS10: Computational Methods in ECE, Lecture 4

(c) 2017 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... */

```

EECS10: Computational Methods in ECE, Lecture 4

(c) 2017 R. Doemer

16



## 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!

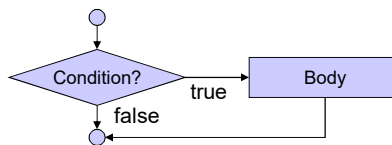
• Example:

```

/* some statements... */
indentation level 0 if (x < 0) {
indentation level 1     printf("%d is negative!", x);
                        /* handle negative values of x... */
                        if (x < -100) {
indentation level 2     printf("%d is too small!", x);
                        /* handle the problem... */
                        } /* fi */
indentation level 1     } /* fi */
indentation level 0 if (x > 0) {
indentation level 1     printf("%d is positive!", x);
                        /* handle positive values of x... */
                        } /* fi */
indentation level 0 /* more statements... */
    
```

## Structured Programming

- Selection: `if` statement
  - Flow chart:



- Example:

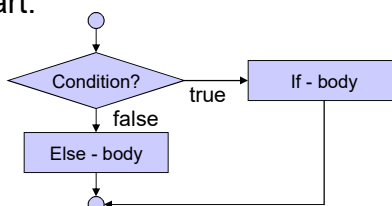
```

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 */
  
```

EECS10: Computational Methods in ECE, Lecture 4

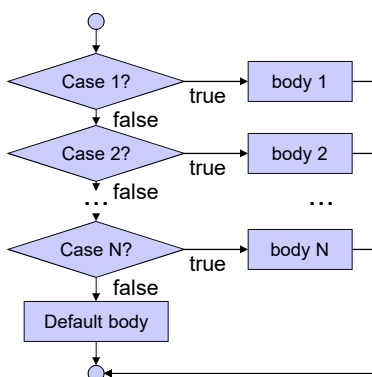
(c) 2017 R. Doemer

19

## 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 4

(c) 2017 R. Doemer

20

## 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 4 (c) 2017 R. Doemer 21

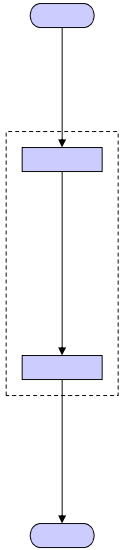
## Structured Program Composition

- Example:
  - Initial flow chart

EECS10: Computational Methods in ECE, Lecture 4 (c) 2017 R. Doemer 22

## Structured Program Composition

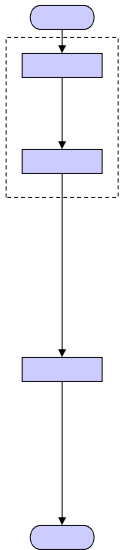
- Example:
  - Sequential composition



EECS10: Computational Methods in ECE, Lecture 4 (c) 2017 R. Doemer 23

## Structured Program Composition

- Example:
  - insertion of another sequential statement



EECS10: Computational Methods in ECE, Lecture 4 (c) 2017 R. Doemer 24

## Structured Program Composition

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

The flowchart shows a sequence of operations starting from a start node (oval), followed by a process node (rectangle), then a decision node (diamond). A dashed box encloses the decision node, a process node to its right, and a process node below the decision node. An arrow points from the decision node to the right-hand process node, and another arrow points from that process node down to the bottom process node. The flow continues from the bottom process node to a final end node (oval).

EECS10: Computational Methods in ECE, Lecture 4 (c) 2017 R. Doemer 25

## Structured Program Composition

- Example:
  - insertion of sequential statement

The flowchart shows a sequence of operations starting from a start node (oval), followed by a process node (rectangle), then a decision node (diamond). A dashed box encloses two process nodes stacked vertically to the right of the decision node. An arrow points from the decision node to the top process node of the dashed box, and another arrow points from the bottom process node of the dashed box down to the process node below the decision node. The flow continues from that process node to a final end node (oval).

EECS10: Computational Methods in ECE, Lecture 4 (c) 2017 R. Doemer 26

## Structured Program Composition

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

The flowchart shows a sequence of operations starting from a start node (oval), followed by a process node (rectangle), a decision node (diamond), and another process node. A dashed box highlights the insertion of an if-else structure: a decision node leads to a process node (the 'if' branch), which then leads to a second decision node. This second decision node branches to a process node (the 'else' branch) and then loops back to the second decision node. After the if-else structure, the flow continues through a final process node and ends at a stop node (oval).

EECS10: Computational Methods in ECE, Lecture 4 (c) 2017 R. Doemer 27

## Structured Program Composition

- Example:
  - insertion of sequential statement

The flowchart shows a sequence of operations starting from a start node (oval), followed by a process node, a decision node, and another process node. A dashed box highlights the insertion of a sequential statement: a decision node leads to a process node, which then leads to a second decision node. This second decision node branches to a process node and then loops back to the second decision node. After the sequential statement, the flow continues through a final process node and ends at a stop node (oval).

EECS10: Computational Methods in ECE, Lecture 4 (c) 2017 R. Doemer 28

## Structured Program Composition

- Example:
  - insertion of sequential statement (twice)

EECS10: Computational Methods in ECE, Lecture 4 (c) 2017 R. Doemer 29

## Structured Program Composition

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

EECS10: Computational Methods in ECE, Lecture 4 (c) 2017 R. Doemer 30

## 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 4

(c) 2017 R. Doemer

31

## 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 4

(c) 2017 R. Doemer

32



## 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 */
```

## 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 */

```

EECS ...

## 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.  
%
```