

# EECS 10: Computational Methods in Electrical and Computer Engineering

## Lecture 10

Rainer Dömer

doemer@uci.edu

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

## Lecture 10: Overview

- Structured Programming
  - Control flow charts
  - Sequential statements
  - Conditional statements
    - `if` statement
    - `if-else` statement
    - `switch` statement
  - Structured Program Composition
  - Example `Grade.c`
  - Example `Grade2.c`

## Structured Programming

- Control flow charts
  - 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 for a program. It begins with a 'Start' terminal, followed by 'Input', 'Compute', and 'Output' process blocks. A decision diamond labeled 'Done?' follows. If the answer is 'No', the flow loops back to the 'Compute' block. If 'Yes', it proceeds to 'Output' and finally to a 'Finish' terminal. Labels on the left side categorize these stages: 'Sequential Execution' for the first four blocks, 'Selection' for the decision diamond, and 'Termination' for the final block.

EECS10: Computational Methods in ECE, Lecture 10 (c) 2008 R. Doemer 3

## 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:

```

graph TD
    In[ ] --> S1[Statement 1]
    S1 --> S2[Statement 2]
    S2 --> S3[Statement 3]
    S3 -.-> Sn[Statement n]
    Sn --> Out[ ]
            
```

The flowchart shows a vertical sequence of rectangular boxes representing 'Statement 1', 'Statement 2', 'Statement 3', and 'Statement n'. Arrows point downwards between them, with a dashed arrow between Statement 3 and Statement n to indicate a continuation of the sequence.

EECS10: Computational Methods in ECE, Lecture 10 (c) 2008 R. Doemer 4

## 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 10

(c) 2008 R. Doemer

5

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

```

EECS10: Computational Methods in ECE, Lecture 10

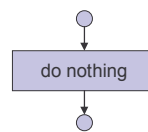
(c) 2008 R. Doemer

6

## Structured Programming

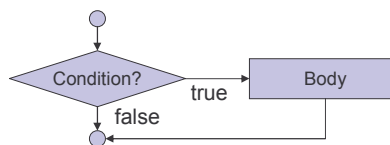
- Empty statement blocks
  - empty compound statement
  - does nothing (no operation, no-op)
  - Example: Flow chart:

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



## Structured Programming

- Selection: **if** statement
  - Flow chart:



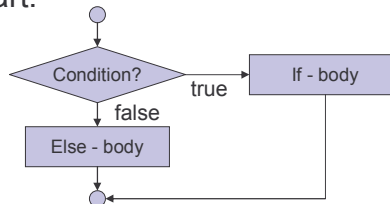
- Example:

```
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 10

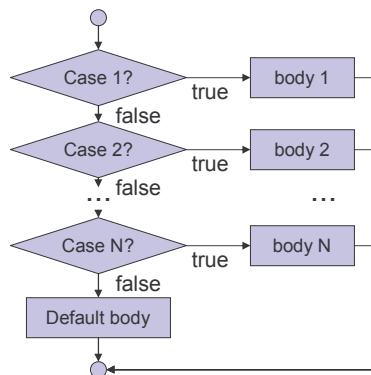
(c) 2008 R. Doemer

9

## 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 10

(c) 2008 R. Doemer

10

## 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 10 (c) 2008 R. Doemer 11

## Structured Program Composition

- Example:
  - Initial flow chart

EECS10: Computational Methods in ECE, Lecture 10 (c) 2008 R. Doemer 12

## Structured Program Composition

- Example:
  - Sequential composition

```

graph TD
    Start([Start]) --> Box1[ ]
    subgraph DashedBox [ ]
        direction TB
        Box1[ ] --> Box2[ ]
    end
    Box2 --> End([End])
    
```

EECS10: Computational Methods in ECE, Lecture 10 (c) 2008 R. Doemer 13

## Structured Program Composition

- Example:
  - insertion of another sequential statement

```

graph TD
    Start([Start]) --> Box1[ ]
    subgraph DashedBox [ ]
        direction TB
        Box1[ ] --> Box2[ ]
    end
    Box2 --> Box3[ ]
    Box3 --> End([End])
    
```

EECS10: Computational Methods in ECE, Lecture 10 (c) 2008 R. Doemer 14

## Structured Program Composition

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

The flowchart shows a sequence of operations: a start node, a process block, a decision diamond, another process block, a third process block, and an end node. A dashed box highlights the decision diamond and the two process blocks that follow it. An arrow from the right side of the decision diamond points to a process block, and another arrow from the bottom of that process block loops back to the bottom of the decision diamond, representing an if-else loop structure.

EECS10: Computational Methods in ECE, Lecture 10 (c) 2008 R. Doemer 15

## Structured Program Composition

- Example:
  - insertion of sequential statement

The flowchart shows a sequence of operations: a start node, a process block, a decision diamond, another process block, a third process block, and an end node. A dashed box highlights the decision diamond and the two process blocks that follow it. An arrow from the right side of the decision diamond points to a process block, which then points to another process block below it. An arrow from the bottom of the second process block loops back to the bottom of the decision diamond, representing a sequential loop structure.

EECS10: Computational Methods in ECE, Lecture 10 (c) 2008 R. Doemer 16



## 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), another process node, and a final process node leading to an end node (oval). A dashed box highlights a section where a decision node is inserted. This decision node has two paths: one leading to a process node and another leading to another process node, both of which then merge back into the main flow.

EECS10: Computational Methods in ECE, Lecture 10 (c) 2008 R. Doemer 17

## 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, another process node, and a final process node leading to an end node (oval). A dashed box highlights a section where a decision node is inserted. This decision node has two paths: one leading to a process node and another leading to a second process node. Both paths then merge back into the main flow.

EECS10: Computational Methods in ECE, Lecture 10 (c) 2008 R. Doemer 18

## 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, and another decision diamond. The second decision diamond branches to a process box, which then leads to a loop of three sequential process boxes enclosed in a dashed box. After the loop, the flow returns to the process box following the second decision diamond, then continues through a final process box to an end node.

EECS10: Computational Methods in ECE, Lecture 10 (c) 2008 R. Doemer 19

## Structured Program Composition

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

The flowchart shows a sequence of operations: a start node, a process box, a decision diamond, a process box, and another decision diamond. The second decision diamond branches to a process box, which then leads to a loop of three sequential process boxes. After the loop, the flow returns to the process box following the second decision diamond, then continues through a final process box to an end node.

EECS10: Computational Methods in ECE, Lecture 10 (c) 2008 R. Doemer 20

## 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 10

(c) 2008 R. Doemer

21

## 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 10

(c) 2008 R. Doemer

22

## 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.  
%
```