

EECS 10: Computational Methods in Electrical and Computer Engineering

Lecture 9

Rainer Dömer

doemer@uci.edu

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

Lecture 9: Overview

- Programming Principles
 - Algorithm
- Structured Programming
 - Control flow charts
 - Sequential statements
 - Conditional statements
 - `if` statement
 - `if-else` statement
 - `switch` statement
 - Structured Program Composition
 - Example `Grade.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*
 - Execution order of statements in the program
- *Program*: Instructions for the computer
 - Formal description in programming language
 - Statements (steps, actions)
 - Control structures (flow of control)

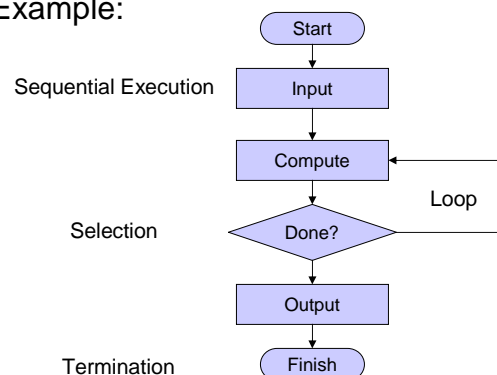
EECS10: Computational Methods in ECE, Lecture 9

(c) 2004 R. Doemer

3

Structured Programming

- Control flow charts
 - Graphical representation of program control flow
 - Example:



EECS10: Computational Methods in ECE, Lecture 9

(c) 2004 R. Doemer

4

Structured Programming

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

```

{
  /* statement 1 */

  /* statement 2 */

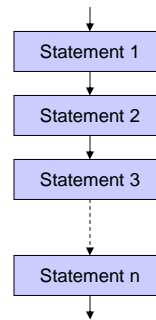
  /* statement 3 */

  /* ... */

  /* statement n */
}

```

Flow chart:



EECS10: Computational Methods in ECE, Lecture 9

(c) 2004 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... */
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 9

(c) 2004 R. Doemer

6

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

EECS10: Computational Methods in ECE, Lecture 9

(c) 2004 R. Doemer

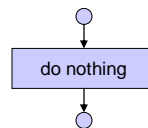
7

Structured Programming

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

```

{
  /* nothing */
}
  
```



EECS10: Computational Methods in ECE, Lecture 9

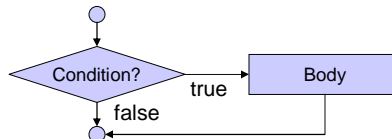
(c) 2004 R. Doemer

8

Structured Programming

- Selection: **if** statement

– Flow chart:



– Example:

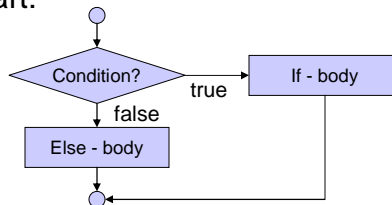
```

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

Structured Programming

- Selection: **if-else** statement

– Flow chart:



– Example:

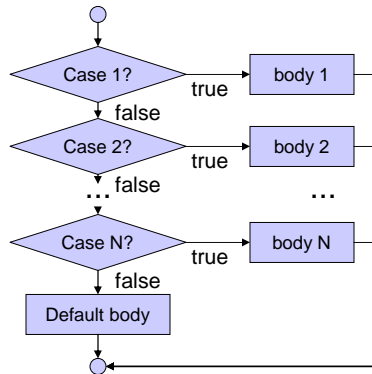
```

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 9

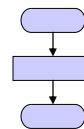
(c) 2004 R. Doemer

11

Structured Program Composition

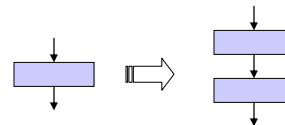
- Initial flow chart

- Start
- Program body
- Finish



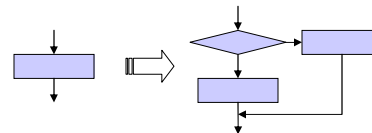
- Statement sequences

- 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 9

(c) 2004 R. Doemer

12

Structured Program Composition

- Example:
 - Initial flow chart

```

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

EECS10: Computational Methods in ECE, Lecture 9 (c) 2004 R. Doemer 13

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 9 (c) 2004 R. Doemer 14

Structured Program Composition

- Example:
 - insertion of another sequential statement

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

EECS10: Computational Methods in ECE, Lecture 9 (c) 2004 R. Doerner 15

Structured Program Composition

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

The flowchart illustrates a program structure with a conditional branch. It starts with an oval representing the start of the program. An arrow points down to a rectangular process box. Below this is a diamond-shaped decision box, also enclosed in a dashed-line rectangle to indicate it is the target for modification. From the decision box, two paths emerge: one goes right to a rectangular process box, and the other goes down to another rectangular process box. An arrow from the right-side process box loops back to the bottom of the decision box. Finally, an arrow from the bottom process box leads to an oval representing the end of the program.

EECS10: Computational Methods in ECE, Lecture 9 (c) 2004 R. Doerner 16

Structured Program Composition

- Example:
 - insertion of sequential statement

The flowchart shows a sequence of operations: a start node, a process box, a decision diamond, another process box, and a final end node. A loop is formed by a process box and a decision diamond. A dashed box highlights the insertion of a new process box between the original process box and the loop's decision diamond. The flow continues from the new process box to the loop's decision diamond, then to the loop's process box, and back to the loop's decision diamond.

EECS10: Computational Methods in ECE, Lecture 9 (c) 2004 R. Doemer 17

Structured Program Composition

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

The flowchart shows a sequence of operations: a start node, a process box, a decision diamond, another process box, and a final end node. A loop is formed by a process box and a decision diamond. A dashed box highlights the insertion of an if-else statement between the original process box and the loop's decision diamond. The if-else statement consists of a decision diamond, two process boxes, and a merge point. The flow continues from the new process box to the if-else decision diamond, then to either of the two process boxes, then to the merge point, and back to the loop's decision diamond.

EECS10: Computational Methods in ECE, Lecture 9 (c) 2004 R. Doemer 18

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 decision diamond, a process box, and an end node. A dashed box highlights the loop body. An arrow from the first decision diamond points to the first process box of the loop body. An arrow from the second decision diamond points to the second process box of the loop body. An arrow from the third decision diamond points to the process box following the loop body.

EECS10: Computational Methods in ECE, Lecture 9 (c) 2004 R. Doemer 19

Structured Program Composition

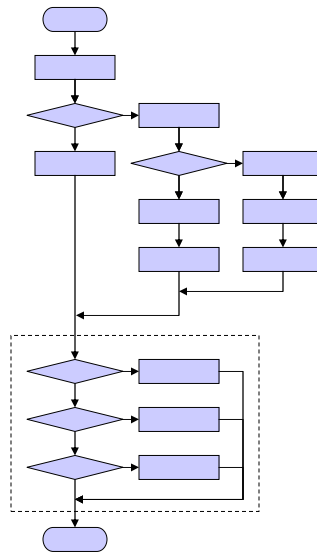
- Example:
 - insertion of sequential statement (twice)

The flowchart is similar to the previous one, but the loop body contains three process boxes instead of two. A dashed box highlights the loop body. An arrow from the first decision diamond points to the first process box of the loop body. An arrow from the second decision diamond points to the second process box of the loop body. An arrow from the third decision diamond points to the third process box of the loop body. An arrow from the fourth decision diamond points to the process box following the loop body.

EECS10: Computational Methods in ECE, Lecture 9 (c) 2004 R. Doemer 20

Structured Program Composition

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



EECS10: Computational Methods in ECE, Lecture 9

(c) 2004 R. Doemer

21

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 9

(c) 2004 R. Doemer

22

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 */
      } /* esle */
    } /* esle */
  } /* esle */
...

```

EECS10: Computational Methods in ECE, Lecture 9

(c) 2004 R. Doemer

23

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 9

(c) 2004 R. Doemer

24

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.
%
```