

EECS 22: Advanced C Programming

Lecture 3 (Tu,Th)

Rainer Dömer

doemer@uci.edu

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

Overview

- Review of the C Programming Language
 - Control Flow Charts
 - Structured Programming
 - Sequential statements
 - Conditional statements
 - Repetition statements
 - Arbitrary jump statements
 - Structured Program Composition
 - Example `Average.c`

Structured Programming

- Control Flow Statements
 - Sequential execution
 - Compound statements
 - Conditional execution
 - `if` statement
 - `if-else` statement
 - `switch` statement
 - Iterative execution
 - `while` loop
 - `do-while` loop
 - `for` loop
 - Unstructured execution
 - `goto` statement

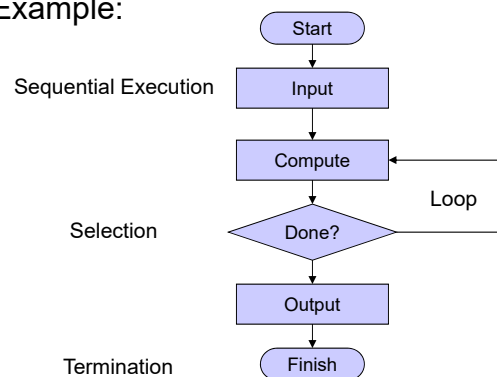
EECS22: Advanced C Programming, Lecture 4

(c) 2017 R. Doemer

3

Structured Programming

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



EECS22: Advanced C Programming, Lecture 4

(c) 2017 R. Doemer

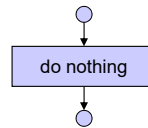
4

Structured Programming

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

Flow chart:

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



EECS22: Advanced C Programming, Lecture 4

(c) 2017 R. Doemer

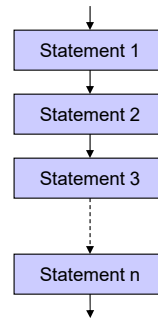
5

Structured Programming

- Compound Statement Blocks
 - Sequence of statements grouped by braces: { }
 - *Sequential* execution of a sequence of statements
- Example:

Flow chart:

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



EECS22: Advanced C Programming, Lecture 4

(c) 2017 R. Doemer

6

Structured Programming

- Compound Statement Blocks
 - Sequence of statements grouped by braces: { }
 - Compound statements may have *local variables!*
- Example:

```

{ /* declarations */
  /* definitions */

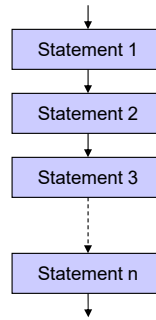
  /* statement 1 */
  /* statement 2 */
  /* statement 3 */

  /* ... */

  /* statement n */
}

```

Flow chart:



EECS22: Advanced C Programming, Lecture 4

(c) 2017 R. Doemer

7

Structured Programming

- Compound Statement Blocks
 - 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... */

```

EECS22: Advanced C Programming, Lecture 4

(c) 2017 R. Doemer

8

Structured Programming

- Compound Statement Blocks
 - 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... */

```

EECS22: Advanced C Programming, Lecture 4

(c) 2017 R. Doemer

9

Structured Programming

- Compound Statement Blocks
 - Sequence of statements grouped by braces: { }
- *Avoid single statements!*
 - Wrapping in braces is highly recommended!
 - Indentation can be misleading! (*C is not Python!*)
- Example:

```

/* some statements... */
if (x < 0)
    printf("%d is negative!", x);

if (x > 0)
    printf("%d is positive!", x);

/* more statements... */

```

EECS22: Advanced C Programming, Lecture 4

(c) 2017 R. Doemer

10

Structured Programming

- Compound Statement Blocks
 - Sequence of statements grouped by braces: { }
- *Avoid single statements!*
 - Wrapping in braces is highly recommended!
 - Indentation can be misleading! (*C is not Python!*)
- Example:

```
/* some statements... */
if (x < 0)
    printf("%d is negative!", x);
    y = sqrt(-x); /* ERROR! */

if (x > 0)
    printf("%d is positive!", x);
    y = sqrt(x); /* ERROR! */

/* more statements... */
```

EECS22: Advanced C Programming, Lecture 4

(c) 2017 R. Doemer

11

Structured Programming

- Compound Statement Blocks
 - Sequence of statements grouped by braces: { }
- *Avoid single statements!*
 - Wrapping in braces is highly recommended!
 - Indentation can be misleading! (*C is not Python!*)
- Example:

```
/* some statements... */
if (x < 0) {
    printf("%d is negative!", x);
    y = sqrt(-x);
} /* fi */

if (x > 0) {
    printf("%d is positive!", x);
    y = sqrt(x);
} /* fi */

/* more statements... */
```

EECS22: Advanced C Programming, Lecture 4

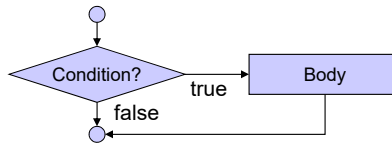
(c) 2017 R. Doemer

12

Structured Programming

- Selection: **if** statement

– Flow chart:



– Example:

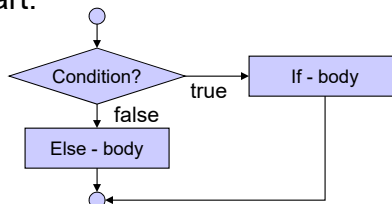
```

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 */
```

EECS22: Advanced C Programming, Lecture 4 (c) 2017 R. Doemer 15

Structured Programming

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

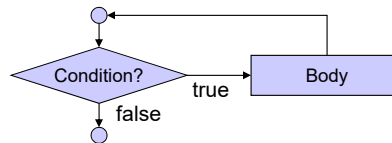

```
switch(LetterGrade)
{ case 'A':
  { printf("Excellent!");
    break; }
  case 'B':
  case 'C':
  case 'D':
    { printf("Passed.");
      break; }
  case 'F':
    { printf("Failed!");
      break; }
  default:
    { printf("Invalid grade!");
      break; }
} /* hctiws */
```

EECS22: Advanced C Programming, Lecture 4 (c) 2017 R. Doemer 16

Structured Programming

- Repetition: **while** loop

- Flow chart:



- Example:

```
int product = 2;
while (product < 1000)
{ product *= 2;
  } /* elihw */
```

- Note:

- The condition is evaluated at the *beginning* of each loop!

EECS22: Advanced C Programming, Lecture 4

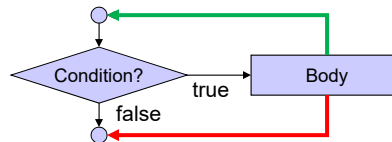
(c) 2017 R. Doemer

17

Structured Programming

- Repetition: **break**/**continue** in **while** loop

- Flow chart:



- Control flow:

- control flow with **break**
- control flow with **continue**

- Note:

- The condition is evaluated at the *beginning* of each loop!

EECS22: Advanced C Programming, Lecture 4

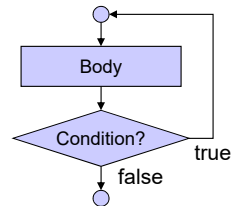
(c) 2017 R. Doemer

18

Structured Programming

- Repetition: **do-while** loop

- Flow chart:



- Example:

```
int product = 2;
do { product *= 2;
    } while (product < 1000);
```

- Note:

- The condition is evaluated at the *end* of each loop!

EECS22: Advanced C Programming, Lecture 4

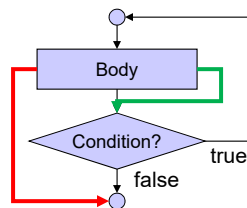
(c) 2017 R. Doemer

19

Structured Programming

- Repetition: **break**/**continue** in **do-while** loop

- Flow chart:



- Control flow:

- control flow with **break**
- control flow with **continue**

- Note:

- The condition is evaluated at the *end* of each loop!

EECS22: Advanced C Programming, Lecture 4

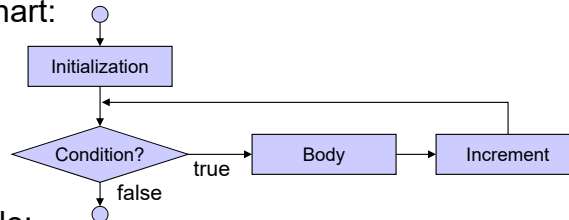
(c) 2017 R. Doemer

20

Structured Programming

- Repetition: **for** loop

– Flow chart:



– Example:

```

for(i = 0; i < 10; i++)
{ printf("i = %d\n", i);
} /* rof */
  
```

– Syntax:

```

for(initialization; condition; increment)
{ body }
  
```

EECS22: Advanced C Programming, Lecture 4

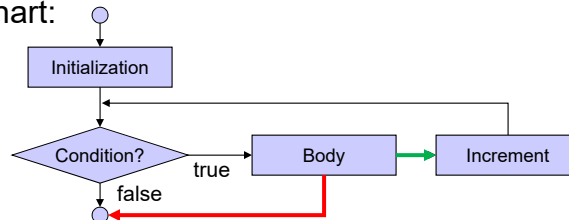
(c) 2017 R. Doemer

21

Structured Programming

- Repetition: **break**/**continue** in **for** loop

– Flow chart:



– Control flow:

→ control flow with **break**

→ control flow with **continue**

– Syntax:

```

for(initialization; condition; increment)
{ body }
  
```

EECS22: Advanced C Programming, Lecture 4

(c) 2017 R. Doemer

22

Arbitrary Control Flow

- Arbitrary jumps: `goto` statement
 - `goto` statement jumps to the specified *labeled* statement (within the same function)

– Example:

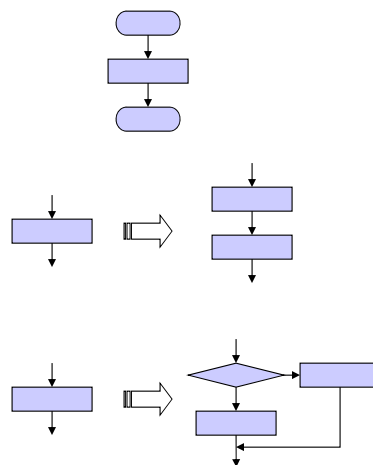
```
begin:  count = 0;
        goto next;
repeat: if (count > 100)
        { goto end; }
next:   count++;
        if (count == 77)
        { goto next; }
        goto repeat;
end:    printf("%d", count);
```

– **Warning:**

- `goto` statement allows *un-structured programming!*
- `goto` statement should be avoided whenever possible!

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



Structured Program Composition

- Example:
 - Initial flow chart

```

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

EECS22: Advanced C Programming, Lecture 4 (c) 2017 R. Doemer 25

Structured Program Composition

- Example:
 - Sequential composition

```

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

EECS22: Advanced C Programming, Lecture 4 (c) 2017 R. Doemer 26

Structured Program Composition

- Example:
 - insertion of another sequential statement

The flowchart shows a vertical sequence of nodes. It starts with an oval at the top, followed by a rectangular box, then another rectangular box, and finally an oval at the bottom. A dashed rectangular box encloses the two middle rectangular boxes. An arrow points from the top oval to the first rectangular box, and another arrow points from the second rectangular box to the bottom oval. The text 'insertion of another sequential statement' is positioned to the left of the dashed box.

EECS22: Advanced C Programming, Lecture 4 (c) 2017 R. Doemer 27

Structured Program Composition

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

The flowchart shows a vertical sequence of nodes. It starts with an oval at the top, followed by a rectangular box, then a diamond-shaped decision node, then another rectangular box, and finally an oval at the bottom. A dashed rectangular box encloses the diamond node, the rectangular box to its right, and the rectangular box below it. An arrow points from the top oval to the first rectangular box, and another arrow points from the second rectangular box to the bottom oval. A third arrow points from the diamond node to the rectangular box on the right, and a fourth arrow points from that box back to the diamond node. The text 'insertion of if - else statement' is positioned to the left of the dashed box.

EECS22: Advanced C Programming, Lecture 4 (c) 2017 R. Doemer 28

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 dashed box highlights a new process node being inserted after the first decision node. An arrow from the decision node points to this new process node, and another arrow from the new process node points back to the original process node, effectively inserting a new step into the flow.

EECS22: Advanced C Programming, Lecture 4 (c) 2017 R. Doemer 29

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 dashed box highlights a new decision node being inserted after the first process node. An arrow from the first decision node points to this new decision node. From the new decision node, two paths emerge: one leading to a process node and another leading to another process node. Both paths then merge back into the original process node, effectively inserting an if-else block into the flow.

EECS22: Advanced C Programming, Lecture 4 (c) 2017 R. Doemer 30

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 first decision diamond points to the first process box of the loop body. An arrow from the second decision diamond points to the process box following the loop body. The flow returns to the start of the loop body after the loop body process boxes.

EECS22: Advanced C Programming, Lecture 4 (c) 2017 R. Doemer 31

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 process box, a loop body (three process boxes), 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 process box following the loop body. The flow returns to the start of the loop body after the loop body process boxes.

EECS22: Advanced C Programming, Lecture 4 (c) 2017 R. Doemer 32

Structured Program Composition

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

EECS22: Advanced C Programming, Lecture 4 (c) 2017 R. Doemer 33

Structured Program Example

- Example **Average.c**
- Task:
 - Compute the average of a set of floating point values
 - The user enters the values consecutively
 - The user enters -1 when done
 - Sentinel-controlled repetition
 - Print the number of values entered and the calculated average
- Caution:
 - The average of 0 values is undefined!

EECS22: Advanced C Programming, Lecture 4 (c) 2017 R. Doemer 34

Structured Program Example

- Average of values: `Average.c` (part 1/3)

```

/* Average.c: compute the average of a set of numbers */
/*
/* author: Rainer Doemer */
/*
/* modifications: */
/* 10/10/04 RD sentinel controlled loop */
/* 10/10/04 RD initial version */

#include <stdio.h>

/* main function */
int main(void)
{
    /* variable definitions */
    int counter;
    double value;
    double total;
    double average;
    ...

```

EECS22: Advanced C Programming, Lecture 4

(c) 2017 R. Doemer

35

Structured Program Example

- Average of values: `Average.c` (part 2/3)

```

...

/* input and computation section */
counter = 0;
total = 0.0;
while (1)
{ printf("Please enter a value (or -1 to quit): ");
  scanf("%lf", &value);
  if (value == -1.0)
  { break;
    } /* fi */
  total += value;
  counter++;
} /* elihw */

...

```

EECS22: Advanced C Programming, Lecture 4

(c) 2017 R. Doemer

36

Structured Program Example

- Average of values: `Average.c` (part 3/3)

```

...

/* computation and output section */
printf("%d values entered.\n", counter);
if (counter >= 1)
    { average = total / counter;
      printf("The average is %f.\n", average);
    } /* fi */

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

/* EOF */

```

Structured Program Example

- Example session: `Average.c`

```

% vi Average.c
% gcc Average.c -o Average -Wall -ansi -std=c99
% ./Average
Please enter a value (or -1 to quit): 2
Please enter a value (or -1 to quit): 3
Please enter a value (or -1 to quit): 4
Please enter a value (or -1 to quit): 5
Please enter a value (or -1 to quit): -1
4 values entered.
The average is 3.500000.
% ./Average
Please enter a value (or -1 to quit): -1
0 values entered.
%

```