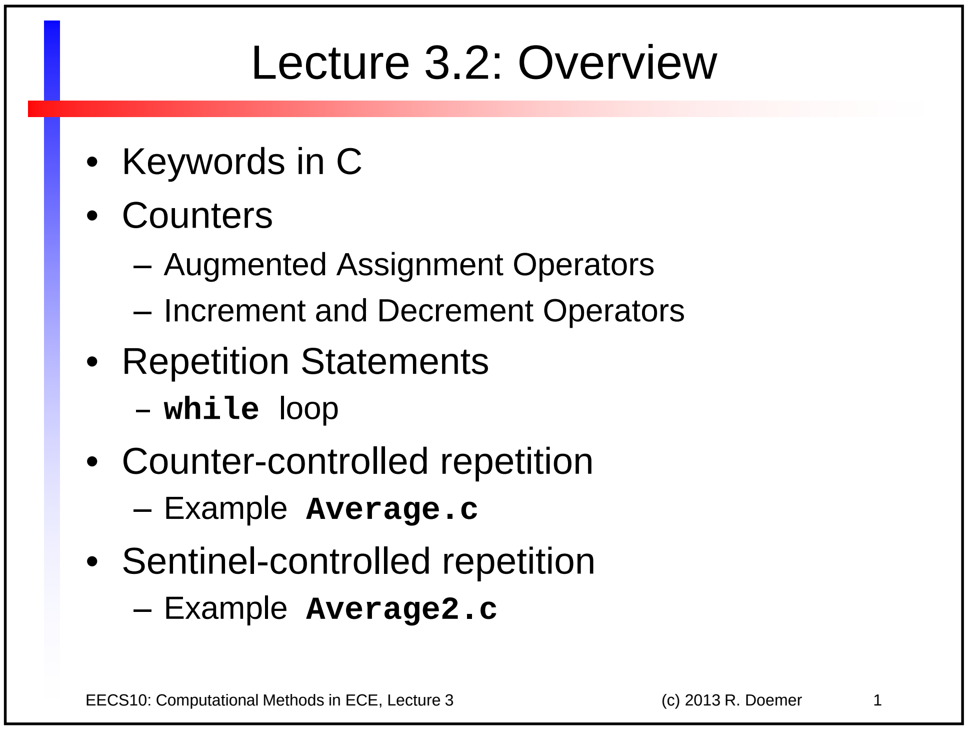
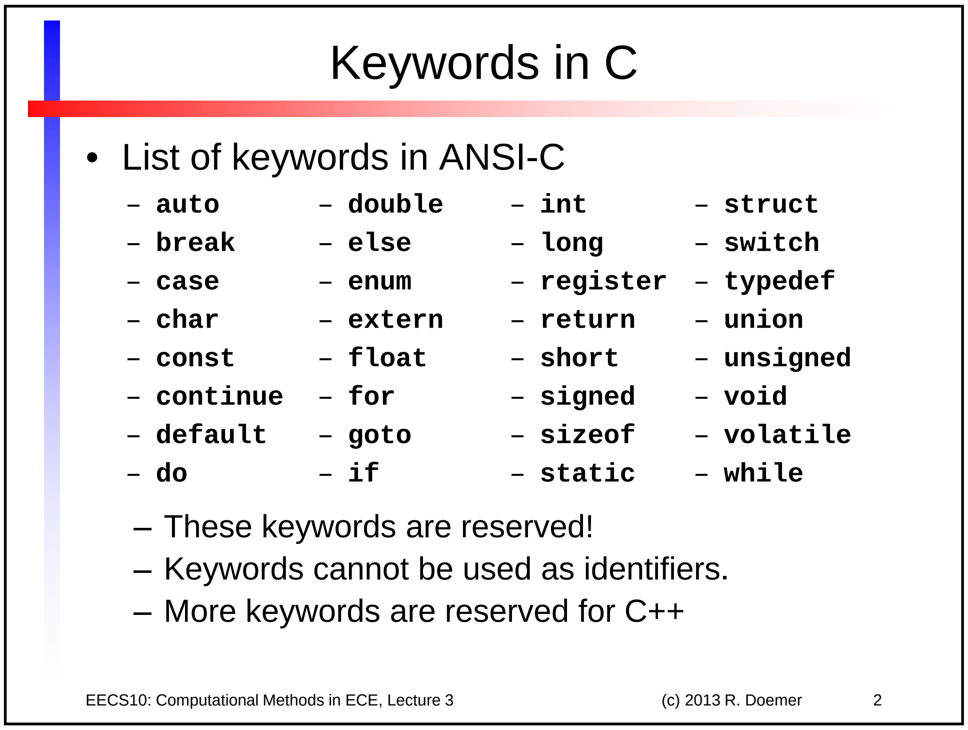


Lecture 3.2: Overview

- Keywords in C
- Counters
  - Augmented Assignment Operators
  - Increment and Decrement Operators
- Repetition Statements
  - **`while`** loop
- Counter-controlled repetition
  - Example **`Average.c`**
- Sentinel-controlled repetition
  - Example **`Average2.c`**



Keywords in C

- List of keywords in ANSI-C

– **auto**	– **double**	– **int**	– **struct**
– **break**	– **else**	– **long**	– **switch**
– **case**	– **enum**	– **register**	– **typedef**
– **char**	– **extern**	– **return**	– **union**
– **const**	– **float**	– **short**	– **unsigned**
– **continue**	– **for**	– **signed**	– **void**
– **default**	– **goto**	– **sizeof**	– **volatile**
– **do**	– **if**	– **static**	– **while**

  - These keywords are reserved!
  - Keywords cannot be used as identifiers.
  - More keywords are reserved for C++

Augmented Assignment Operators

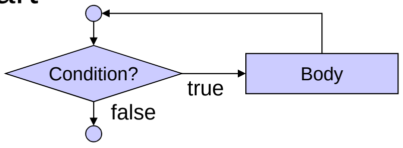
- Assignment operator: **`=`**
  - evaluates right-hand side
  - assigns result to left-hand side
- Augmented assignment operators: **`+=`**, **`*=`**, ...
  - evaluates right-hand side as temporary result
  - applies operation to left-hand side and temporary result
  - assigns result of operation to left-hand side
- Example: Counter
  - **`int c = 0;`** `/*` counter starting from 0 `*/`
  - **`c = c + 1;`** `/*` counting by regular assignment `*/`
  - **`c += 1;`** `/*` counting by augmented assignment `*/`
- Augmented assignment operators:
  - **`+=`**, **`-=`**, **`*=`**, **`/=`**, **`%=`**, **`<<=`**, **`>>=`**, **`||=`**, **`&&=`**

Increment and Decrement Operators

- Counting in steps of one
  - increment (add 1)
  - decrement (subtract 1)
- C provides special operators
  - increment operator: **`++`**
    - **`count++`** post-increment (**`count += 1`**)
    - **`++count`** pre-increment (**`count += 1`**)
  - decrement operator: **`--`**
    - **`count--`** post-decrement (**`count -= 1`**)
    - **`--count`** pre-decrement (**`count -= 1`**)
  - *pre-* increment/decrement
    - value returned is the incremented/decremented (new) value
  - *post-* increment/decrement
    - value returned is the original (old) value

Repetition Statements

- Repetition (aka. iteration, loop)
  - repeated execution of a block of statements
  - counter-controlled
    - counter determines number of repetitions (often predefined at compile time)
  - sentinel-controlled
    - sentinel condition determines number of repetitions (usually determined at run time)
- Control flow chart



Repetition Statements

- **`while`** loop
  - Control flow statement for repetition (iteration)
    - Repeats execution depending on a specified condition
  - Example:

```
int product = 2;
while (product < 1000)
   { product *= 2; }
printf(“Product is %d”, product);
```

  - Syntax: **`while`** construct consists of

• keyword	**`while`**
• condition	expression evaluated to true or false
• body	statement block

  - Semantics: the body is repeatedly executed as long as the condition evaluates to true
    - the condition is evaluated at the *beginning* of each loop

Example Program

- 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  initial version                           */

#include <stdio.h>

/* main function */

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

Example Program

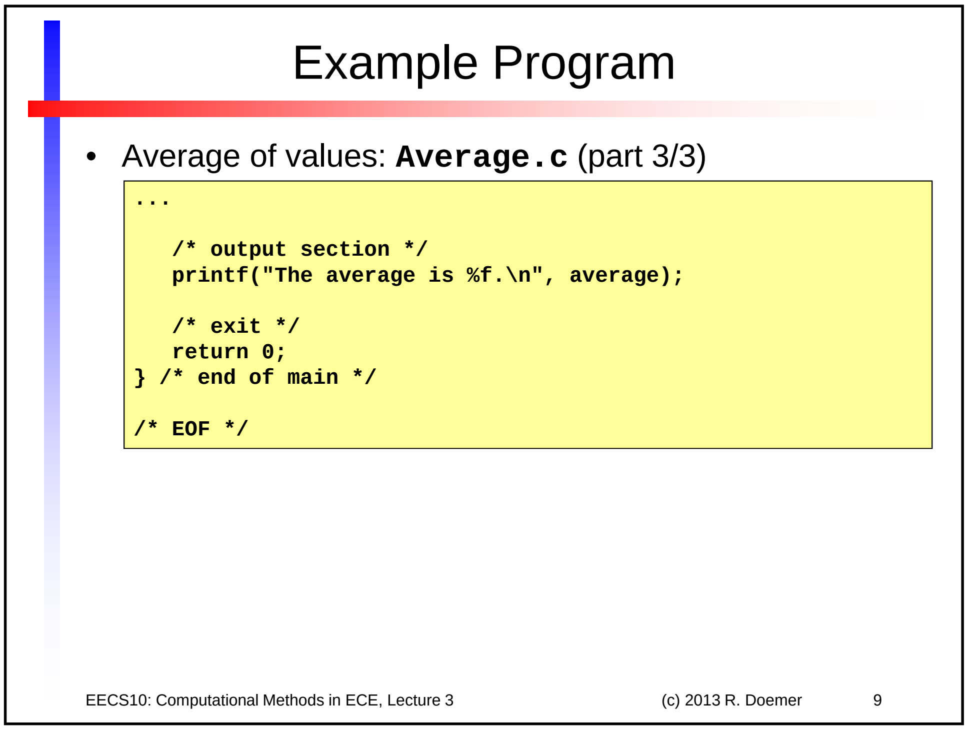
- Average of values: **`Average.c`** (part 2/3)

```
...

   /* input and computation section */
   counter = 1;
   total = 0.0;
   while (counter <= 10)
      { printf("Please enter value %d: ", counter);
        scanf("%lf", &value);
        total += value;
        counter++;
       } /* elihw */

   /* computation section */
   average = total / 10.0;

...
```



Example Program

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

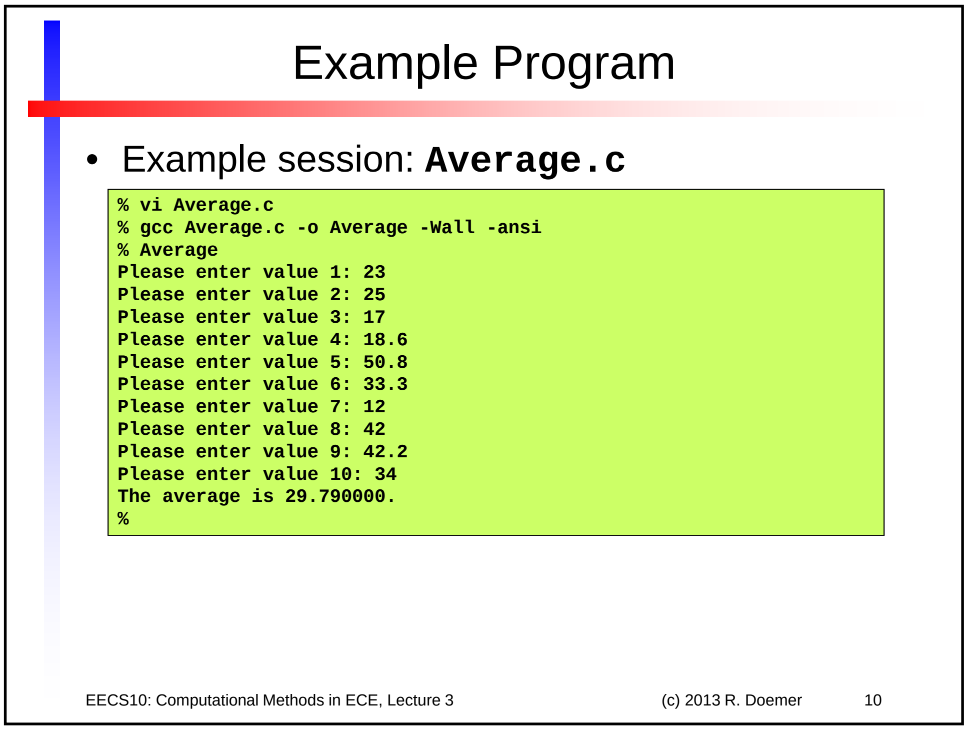
```
...

   /* output section */
   printf("The average is %f.\n", average);

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

/* EOF */
```

EECS10: Computational Methods in ECE, Lecture 3		(c) 2013 R. Doemer		9



Example Program

- Example session: **`Average.c`**

```
% vi Average.c
% gcc Average.c -o Average -Wall -ansi
% Average
Please enter value 1: 23
Please enter value 2: 25
Please enter value 3: 17
Please enter value 4: 18.6
Please enter value 5: 50.8
Please enter value 6: 33.3
Please enter value 7: 12
Please enter value 8: 42
Please enter value 9: 42.2
Please enter value 10: 34
The average is 29.790000.
%
```

EECS10: Computational Methods in ECE, Lecture 3		(c) 2013 R. Doemer		10

Repetition Statements

- Explicit control flow in loops
  - **`break`** statement
    - exits the innermost loop
  - **`continue`** statement
    - jump back to the beginning of the innermost loop
- Example:

```
int i = 0;
int s = 0;
while (1)            /* “endless” loop */
  { i++;
    if (i > 10)
       { break; }    /* exit the loop */
    if (i % 2 == 1)
       { continue; }/* next iteration */
    s += i;
   } /* elihw */
printf(“%d”, s);
```

Example Program

- Average of values: **`Average2.c`** (part 1/3)

```
/* Average2.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;
...
```

Example Program

- Average of values: **`Average2.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 */

...
```

Example Program

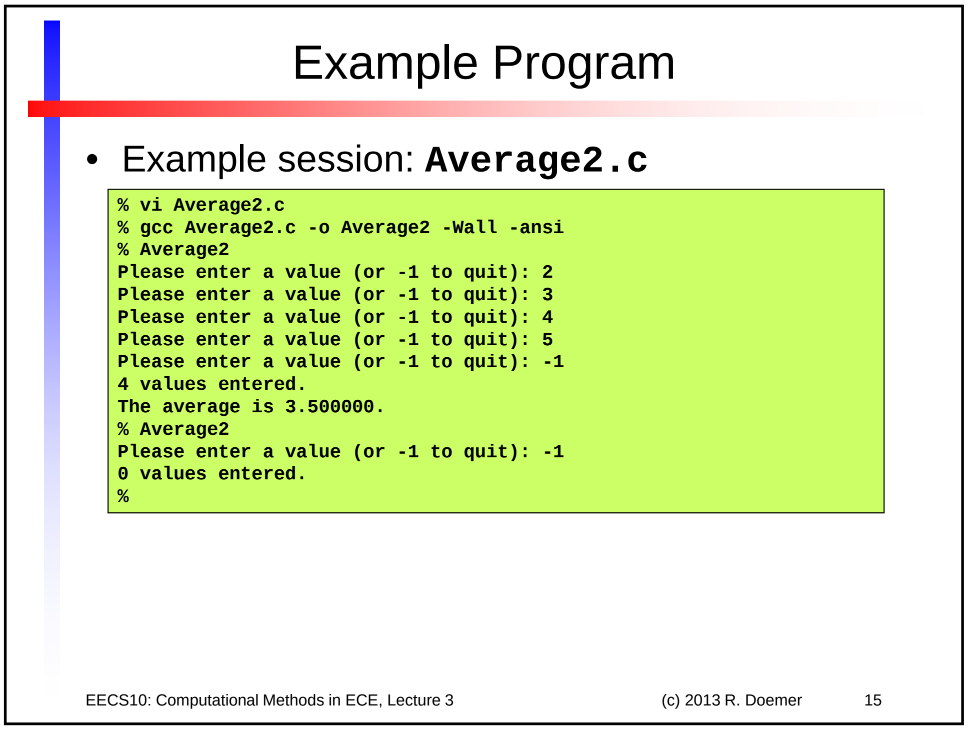
- Average of values: **`Average2.c`** (part 3/3)

```
...

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

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

/* EOF */
```



Example Program

- Example session: **`Average2.c`**

```
% vi Average2.c
% gcc Average2.c -o Average2 -Wall -ansi
% Average2
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.
% Average2
Please enter a value (or -1 to quit): -1
0 values entered.
%
```

EECS10: Computational Methods in ECE, Lecture 3   (c) 2013 R. Doemer   15