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

- Midterm 1 Review Quiz
  - Top 5 most “difficult” questions
- Formatted output
  - Formatting of integral values
  - Formatting of floating-point values
  - Example **`Formatting.c`**
- Programming Principles
  - Algorithm
  - Control flow

Midterm 1 Review Quiz

- Top 5 most "difficult" questions:
  - Rank 5: Question 27 (35.3% incorrect answers)
- Prime number test: Iterate over $2 \le i < x$ to find a divisor of x. What should go into box in line 4? (1 pt.)
  a) `i = 0;`
  b) `i = 1;`
  c) `i = 2;`
  d) `i = x;`
  e) `x = 0;`

```
int x, i;
printf("Please input a number: ");
scanf("%d", &x);
initialize variable i
while(i < x)
{ if(x % i == 0)
  { printf("%d is not prime\n", x);
    break;
  }
  i++;
}
if( none of the i is a divisor of x )
  { printf("%d is prime\n", x);
  }
```


Midterm 1 Review Quiz

- Top 5 most "difficult" questions:
  - Rank 5: Question 27 (35.3% incorrect answers)
- Prime number test: Iterate over $2 \le i < x$ to find a divisor of x. What should go into box in line 4? (1 pt.)
  a) `i = 0;`
  b) `i = 1;`
  c) `i = 2;`
  d) `i = x;`
  e) `x = 0;`

```
int x, i;
printf("Please input a number: ");
scanf("%d", &x);
initialize variable i
while(i < x)
{ if(x % i == 0)
  { printf("%d is not prime\n", x);
    break;
  }
  i++;
}
if( none of the i is a divisor of x )
  { printf("%d is prime\n", x);
  }
```

EECS10: Computational Methods in ECE, Lecture 9 (c) 2010 R. Doemer 4

Midterm 1 Review Quiz

- Top 5 most "difficult" questions:
  - Rank 4: Question 25 (50.9% incorrect answers)
- Which of the following program fragments will *not* terminate? (Check all that apply! 2 pts.)

a)
```
int a = 1;
while(a < 1000000)
   { a++; }
```

b)
```
int a = 0;
while(a < 1000)
   { a = a * 3; }
```

c)
```
int a = 1;
while(a == 1)
   { a = a % 10; }
```

d)
```
int a = 10;
while(a > 0)
   { a = a / 3; }
```

e)
```
int a = 1;
while(a < 1000)
   { a = a << 1; }
```

EECS10: Computational Methods in ECE, Lecture 9 (c) 2010 R. Doemer 5

Midterm 1 Review Quiz

- Top 5 most "difficult" questions:
  - Rank 4: Question 25 (50.9% incorrect answers)
- Which of the following program fragments will *not* terminate? (Check all that apply! 2 pts.)

a)
```
int a = 1;
while(a < 1000000)
   { a++; }
```

b)
```
int a = 0;
while(a < 1000)
   { a = a * 3; }
```

c)
```
int a = 1;
while(a == 1)
   { a = a % 10; }
```

d)
```
int a = 10;
while(a > 0)
   { a = a / 3; }
```

e)
```
int a = 1;
while(a < 1000)
   { a = a << 1; }
```

EECS10: Computational Methods in ECE, Lecture 9 (c) 2010 R. Doemer 6

Midterm 1 Review Quiz

- Top 5 most "difficult" questions:
  - Rank 3: Question 12 (63.1% incorrect answers)
- Which of the following C expressions yield the same result?
  (Check all that apply! 2 pts.)

a) `4 << 8 % 5 / 2`
b) `(4 << 8) % 5 / 2`
c) `4 << 8 % (5 / 2)`
d) `(4 << 8 % 5) / 2`
e) `4 << (8 % 5) / 2`

Midterm 1 Review Quiz

- Top 5 most "difficult" questions:
  - Rank 3: Question 12 (63.1% incorrect answers)
- Which of the following C expressions yield the same result?
  (Check all that apply! 2 pts.)

a) `4 << 8 % 5 / 2` **(8)**
b) `(4 << 8) % 5 / 2` **(2)**
c) `4 << 8 % (5 / 2)` **(4)**
d) `(4 << 8 % 5) / 2` **(16)**
e) `4 << (8 % 5) / 2` **(8)**

Midterm 1 Review Quiz

- Top 5 most "difficult" questions:
  - Rank 2: Question 13 (64.3% incorrect answers)
- What is the output of the following C program fragment (1 pt.)

```
int i1 = 5, i2 = 2, i;
float f1 = 5, f2 = 2, f;
i = i1 / i2;
f = (int)(f1 / f2);
printf("i = %d, f = %f", i, f);
```

a) **i = 2, f = 2**
b) **i = 1, f = 2**
c) **i = 2, f = 2.00000**
d) **i = 2.00000, f = 2.50000**
e) **i = 2, f = 2.50000**

Midterm 1 Review Quiz

- Top 5 most "difficult" questions:
  - Rank 2: Question 13 (64.3% incorrect answers)
- What is the output of the following C program fragment (1 pt.)

```
int i1 = 5, i2 = 2, i;
float f1 = 5, f2 = 2, f;
i = i1 / i2;
f = (int)(f1 / f2);
printf("i = %d, f = %f", i, f);
```

a) i = 2, f = 2
b) i = 1, f = 2
c) **i = 2, f = 2.00000**
d) i = 2.00000, f = 2.50000
e) i = 2, f = 2.50000

Midterm 1 Review Quiz

- Top 5 most "difficult" questions:
  - Rank 1: Question 30 (75.1% incorrect answers)
- Prime number test: Iterate over $2 \le i < x$ to find a divisor of x. What should go into box in line 12? (1 pt.)
  a) `x / i == 0`
  b) `x < i`
  c) `i / x == 0`
  d) `i + 1 == x`
  e) `i == x`

```
int x, i;
printf("Please input a number: ");
scanf("%d", &x);
initialize variable i
while(i < x)
{ if(x % i == 0)
  { printf("%d is not prime\n", x);
    break;
  }
  i++;
}
if( none of the i is a divisor of x )
  { printf("%d is prime\n", x);
  }
```


Midterm 1 Review Quiz

- Top 5 most "difficult" questions:
  - Rank 1: Question 30 (75.1% incorrect answers)
- Prime number test: Iterate over $2 \le i < x$ to find a divisor of x. What should go into box in line 12? (1 pt.)
  a) `x / i == 0`
  b) `x < i`
  c) `i / x == 0`
  d) `i + 1 == x`
  e) `i == x`

```
int x, i;
printf("Please input a number: ");
scanf("%d", &x);
initialize variable i
while(i < x)
{ if(x % i == 0)
  { printf("%d is not prime\n", x);
    break;
  }
  i++;
}
if( none of the i is a divisor of x )
  { printf("%d is prime\n", x);
  }
```

EECS10: Computational Methods in ECE, Lecture 9 (c) 2010 R. Doemer 12

Formatted Output

- Formatted output using **`printf()`**
  - standard format specifiers for integral values
    - **`unsigned long long`** **`%llu`**
    - **`long long`** **`%lld`**
    - **`unsigned long`** **`%lu`**
    - **`long`** **`%ld`**
    - **`unsigned int`** **`%u`**
    - **`int`** **`%d`**
    - **`short`** **`%hd`**
  - standard format specifiers for floating point values
    - **`long double`** **`%Lf`**
    - **`double`** **`%f`**
    - **`float`** **`%f`**

EECS10: Computational Methods in ECE, Lecture 9 (c) 2010 R. Doemer 13

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 9 (c) 2010 R. Doemer 14

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**)

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);
...
```

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 |%%12.4f|: |%12.4f|\n",
                                                   123.456);
  printf("123.456 formatted as |%%12.4e|: |%12.4e|\n",
                                                   123.456);
  printf("123.456 formatted as |%%12.4g|: |%12.4g|\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 |%12.4f|: |    123.4560|
123.456 formatted as |%12.4e|: |  1.2346e+02|
123.456 formatted as |%12.4g|: |       123.5|
%
```

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)

Control Flow

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