

# 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 10.1: Overview

- Course Administration
  - Final course evaluation
- File Processing
  - Introduction
  - Standard input and output streams
  - File streams, I/O
  - Standard library functions in `stdio.h`
  - Program example `PhotoLab.c`

## Course Administration

- Final Course Evaluation
  - Open this week
  - July 19, 2014, through Sunday, July 27, 2014
  - Online via EEE Evaluation application
- Mandatory Evaluation of Course and Instructor
  - Voluntary
  - Anonymous
  - Very valuable
    - Help to improve this class!
- Please spend 5 minutes!

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

3

## File Processing

- Introduction
  - Up to now, all data processed is available only during program run time
    - At program completion, all data is lost
  - *Persistent data* is stored even after a program exits
  - Persistent data is stored in files...
    - ... on the harddisk
    - ... on a removable disk (CD, memory stick, ...)
    - ... on a tape, ...
  - Input and output from/to files is organized as *I/O streams*

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

4

## File Processing

- I/O Streams
  - Standard I/O streams (opened by the system)
    - `stdin`      standard input stream (i.e. `scanf()`)
    - `stdout`     standard output stream (i.e. `printf()`)
    - `stderr`    standard error stream (i.e. `perror()`)
  - File I/O streams (explicitly opened by a program)
    - Open a file            `fopen()`
    - Write data to a file `fprintf()`, `fputs()`, etc.
    - Read data from a file `fscanf()`, `fgets()`, etc.
    - Close a file            `fclose()`
  - In C, all I/O functions are ...
    - ... declared in header file `stdio.h`
    - ... implemented in the standard C library

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

5

## Standard I/O Functions

- Functions declared in `stdio.h` (part 1/4)
  - `int printf(const char *fmt, ...);`
  - `int scanf(const char *fmt, ...);`
    - formatted output/input to/from stream `stdin/stdout`
  - `int sprintf(char *s, const char *fmt, ...);`
  - `int sscanf(const char *s, const char *fmt, ...);`
    - formatted output/input to/from a string `s`
  - `int getchar(void);`
  - `int putchar(int c);`
    - input/output of a single character to/from stream `stdin/stdout`
  - `char *gets(char *s);`
  - `int puts(const char *s);`
    - input/output of strings to/from stream `stdin/stdout`

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

6

## Standard I/O Functions

- Functions declared in `stdio.h` (part 2/4)
  - `typedef __FILE FILE;`
    - opaque type for a file handle
  - `FILE *fopen(const char *n, const char *m);`
    - open file named `n` for input ("`r`"), output ("`w`"), or append ("`a`")
    - returns a file handle, or `NULL` in case of an error
  - `int fclose(FILE *f);`
    - closes an open file handle
  - `int fprintf(FILE *f, const char *fmt, ...);`
  - `int fscanf(FILE *f, const char *fmt, ...);`
  - `int fgetc(FILE *f);`
  - `char *fgets(char *s, int n, FILE *f);`
  - `int fputc(int c, FILE *f);`
  - `int fputs(const char *s, FILE *f);`
    - input/output functions from/to stream `f`
  - `int fflush(FILE *f);`
    - flushes any unwritten data from a buffer into the file

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

7

## Standard I/O Functions

- Functions declared in `stdio.h` (part 3/4)
  - `typedef unsigned int size_t;`
    - type for size of a block of memory (number of bytes)
  - `size_t fread(void *p, size_t s, size_t n, FILE *f);`
    - binary input to memory location `p` for `n` times `s` bytes from file `f`
  - `size_t fwrite(const void *p, size_t s, size_t n, FILE *f);`
    - binary output from memory location `p` for `n` times `s` bytes to file `f`
  - `long ftell(FILE *f);`
    - return the current position in file `f` (from beginning)
  - `int fseek(FILE *f, long pos, int w);`
    - move to position `pos` in file `f` (from beginning/current pos/end)
  - `void rewind(FILE *f);`
    - move to beginning of file `f`
  - `int feof(FILE *f);`
    - check if end of file `f` is reached

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

8

## Standard I/O Functions

- Functions declared in `stdio.h` (part 4/4)
  - `int ferror(FILE *f);`
    - returns the current error status for file `f`
  - `void perror(const char *prg);`
    - print current error for program `prg` to stream `stderr`
  - `int remove(const char *filename);`
    - delete file `filename`
  - `int rename(const char *old, const char *new);`
    - rename file `old` to new name `new`

## File Processing

- Program example: **PhotoLab**
  - Digital image manipulation
    - Read an image from a file
    - Manipulate the image in memory
    - Write the modified image to file
  - Portable Pixel Map (PPM) file format
    - Simple uncompressed file format for color images
    - Header section (including picture width, height)
    - Data section (pixel intensities for red, green, and blue)

```
P6
810 450
255
RGBRGBRGB...
```

## File Processing

- Program example: PhotoLab.c (part 1/10)

```

/*****
/* PhotoLab.c: final assignment for EECS 10 in Summer'14 */
/*
/* modifications: (most recent first)
/* 07/22/14 RD adjusted for lecture usage
*****/

#include <stdio.h>
#include <stdlib.h>

/** global definitions */

#define WIDTH 810 /* image width */
#define HEIGHT 450 /* image height */
#define SLEN 80 /* max. string length */

...

```

## File Processing

- Program example: PhotoLab.c (part 2/10)

```

...
/** function definitions */

/* write the RGB image to a PPM file */
/* (return 0 for success, >0 for error) */
int WriteImage(char Filename[SLEN],
               unsigned char R[WIDTH][HEIGHT],
               unsigned char G[WIDTH][HEIGHT],
               unsigned char B[WIDTH][HEIGHT])
{
    FILE *File;
    int x, y;
    File = fopen(Filename, "w");
    if (!File)
    {
        printf("\nCannot open file \"%s\"!\n", Filename);
        return(1);
    }
    ...
}

```

## File Processing

- Program example: PhotoLab.c (part 3/10)

```

...
fprintf(File, "P6\n");
fprintf(File, "%d %d\n", WIDTH, HEIGHT);
fprintf(File, "255\n");
for(y=0; y<HEIGHT; y++)
{
    for(x=0; x<WIDTH; x++)
    {
        fputc(R[x][y], File);
        fputc(G[x][y], File);
        fputc(B[x][y], File);
    }
}
if (ferror(File))
{
    printf("\nFile error while writing to file!\n");
    return(2);
}
fclose(File);
return(0); /* success! */
} /* end of WriteImage */
...

```

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

13

## File Processing

- Program example: PhotoLab.c (part 4/10)

```

...
/* read an RGB image from a PPM file */
/* (return 0 for success, >0 for error) */

int ReadImage(char Filename[SLEN],
              unsigned char R[WIDTH][HEIGHT],
              unsigned char G[WIDTH][HEIGHT],
              unsigned char B[WIDTH][HEIGHT])
{
    FILE *File;
    char Type[SLEN];
    int Width, Height, MaxValue, x, y;

    File = fopen(Filename, "r");
    if (!File)
    {
        printf("\nCannot open file \"%s\"!\n", Filename);
        return(1);
    }
}
...

```

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

14

## File Processing

- Program example: PhotoLab.c (part 5/10)

```

...
fscanf(File, "%79s", Type);
if (Type[0] != 'P' || Type[1] != '6' || Type[2] != 0)
{
    printf("\nUnsupported file format!\n");
    return(2);
}
fscanf(File, "%d", &Width);
if (Width != WIDTH)
{
    printf("\nUnsupported image width %d!\n", Width);
    return(3);
}
fscanf(File, "%d", &Height);
if (Height != HEIGHT)
{
    printf("\nUnsupported image height %d!\n", Height);
    return(4);
}
...

```

## File Processing

- Program example: PhotoLab.c (part 6/10)

```

...
fscanf(File, "%d", &MaxValue);
if (MaxValue != 255)
{
    printf("\nUnsupported maximum %d!\n", MaxValue);
    return(5);
}
if ('\n' != fgetc(File))
{
    printf("\nCarriage return expected!\n");
    return(6);
}
for(y=0; y<HEIGHT; y++)
{
    for(x=0; x<WIDTH; x++)
    {
        R[x][y] = fgetc(File);
        G[x][y] = fgetc(File);
        B[x][y] = fgetc(File);
    }
}
...

```



## File Processing

- Program example: PhotoLab.c (part 7/10)

```

...
    if (ferror(File))
    {   printf("\nFile error while reading from file!\n");
        return(7);
    }
    fclose(File);
    return(0); /* success! */
} /* end of ReadImage */
...

```

## File Processing

- Program example: PhotoLab.c (part 8/10)

```

...
/* modify the image... ;- ) */

void ModifyImage(unsigned char R[WIDTH][HEIGHT],
                unsigned char G[WIDTH][HEIGHT],
                unsigned char B[WIDTH][HEIGHT])
{   int x, y;

    for(y=0; y<HEIGHT; y++)
    {   for(x=0; x<WIDTH; x++)
        {
            B[x][y] = (R[x][y] + G[x][y] + B[x][y]) / 5;
            R[x][y] = (unsigned char) (B[x][y]*1.6);
            G[x][y] = (unsigned char) (B[x][y]*1.6);
        }
    }

} /* end of ModifyImage */
...

```

## File Processing

- Program example: PhotoLab.c (part 9/10)

```

...
/** main program */

int main(void)
{
    /* image data */
    unsigned char R[WIDTH][HEIGHT];
    unsigned char G[WIDTH][HEIGHT];
    unsigned char B[WIDTH][HEIGHT];
    /* file name */
    char Filename[SLEN];

    ...

```

## File Processing

- Program example: PhotoLab.c (part 10/10)

```

...
printf("Enter input file name: ");
scanf("%79s", Filename);
if (ReadImage(Filename, R,G,B) != 0)
{ exit(10);}

/* modify the image */
ModifyImage(R, G, B);

printf("Enter output file name: ");
scanf("%79s", Filename);
if (WriteImage(Filename, R,G,B) != 0)
{ exit(10); }

return 0;
} /* end of main */

/* EOF */

```

## File Processing

- Example session: `PhotoLab.c`

```
% vi PhotoLab.c
% gcc PhotoLab.c -o PhotoLab -Wall -ansi
% PhotoLab
Enter input file name: WorldCup2014.ppm
Enter output file name: WorldCup1954.ppm
%
```

WorldCup2014.ppm



WorldCup1954.ppm



## Lecture 10.2: Overview

- Translation Units
  - Introduction
  - Compiler components
    - Preprocessor
    - Compiler
    - Linker
  - Modules
  - Program example `PhotoLab2`
    - Module `FileIO`
    - Module `Age`
    - Module `Main`

## Translation Units

- Introduction
  - C compilation process is a sequence of phases
    1. Preprocessing (handle # directives)
    2. Scanning and parsing (generate internal data structure)
    3. Instruction generation (emit stream of CPU instructions)
    4. Assembly (generate binary object file)
    5. Linking (combine objects into executable file)
  - C compiler consists of separate components
    - Preprocessor (processes # directives)
    - Compiler (compiles and assembles code)
    - Linker (processes object files and libraries)

EECS10: Computational Methods in ECE, Lecture 10

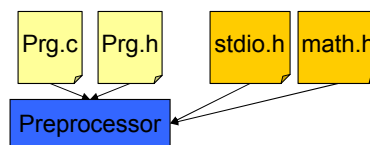
(c) 2014 R. Doemer

23

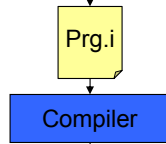
## Translation Units

- Compilation Phases

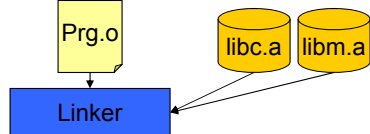
- Source code
  - Program files
  - Header files



- Preprocessed file



- Object files
- Library files



- Executable file

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

24

## Translation Units

- Source files
  - Header files: **Program.h**
    - Inclusion of required header files
    - Definitions of exported constants
    - Declarations of exported global variables
    - Declarations of exported functions
  - Program files: **Program.c**
    - Inclusion of required header files
    - Declaration and definition of local variables
    - Declaration and definition of local functions
    - Definitions of exported global variables
    - Definitions of exported functions

## Translation Units

- C Preprocessor
  - preprocesses source files
  - handles # directives
- Preprocessing Directives
  - Constant definition
  - Macro definition
  - Header file inclusion
  - Conditional compilation

```
#define WIDTH 810
```

```
#define ABS(x) (x>0 ? x : -x)
```

```
#include <stdio.h>
```

```
#define DEBUG /* comment out to turn debugging off */
...
#ifdef DEBUG
printf("value of x is now %d\n", x);
#endif
```

## Translation Units

- Object files
  - **Program.o**
    - Compiled object code of source file **Program.c**
    - Use option `-c` in GNU compiler call to create object files  
`gcc -c Program.c -o Program.o -Wall -ansi`
  - **Library.a**
    - Archive of compiled object files
- Executable file
  - **Program**
    - Object files and libraries linked together into a complete file ready for execution
    - GNU compiler recognizes object files by `.o` suffix, so object files and libraries require no special option  
`gcc Program.o -lc -lm -o Program`

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

27

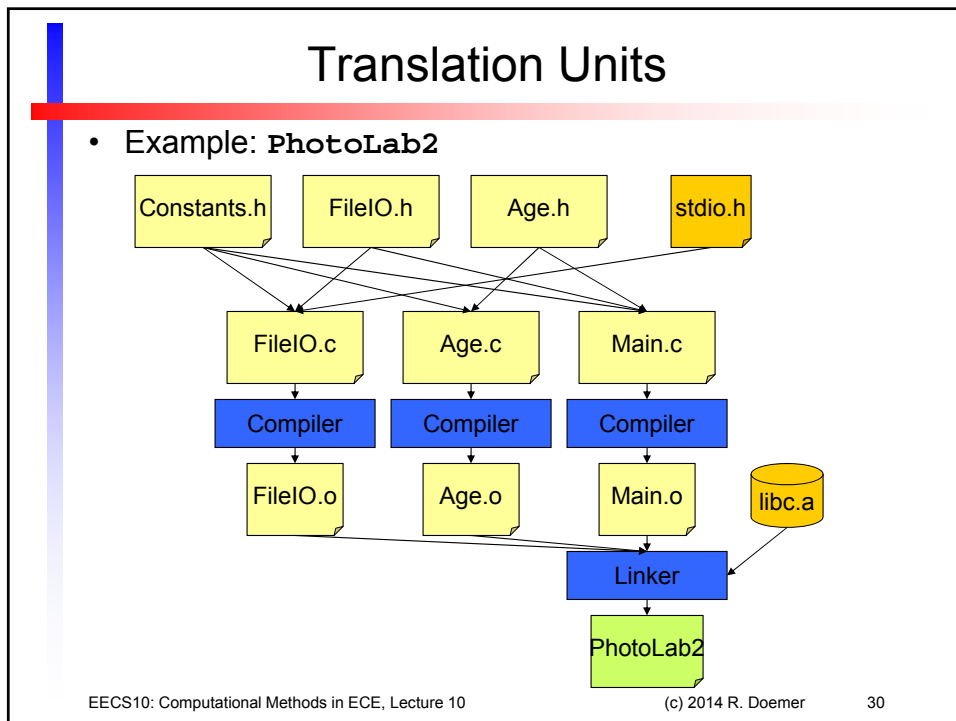
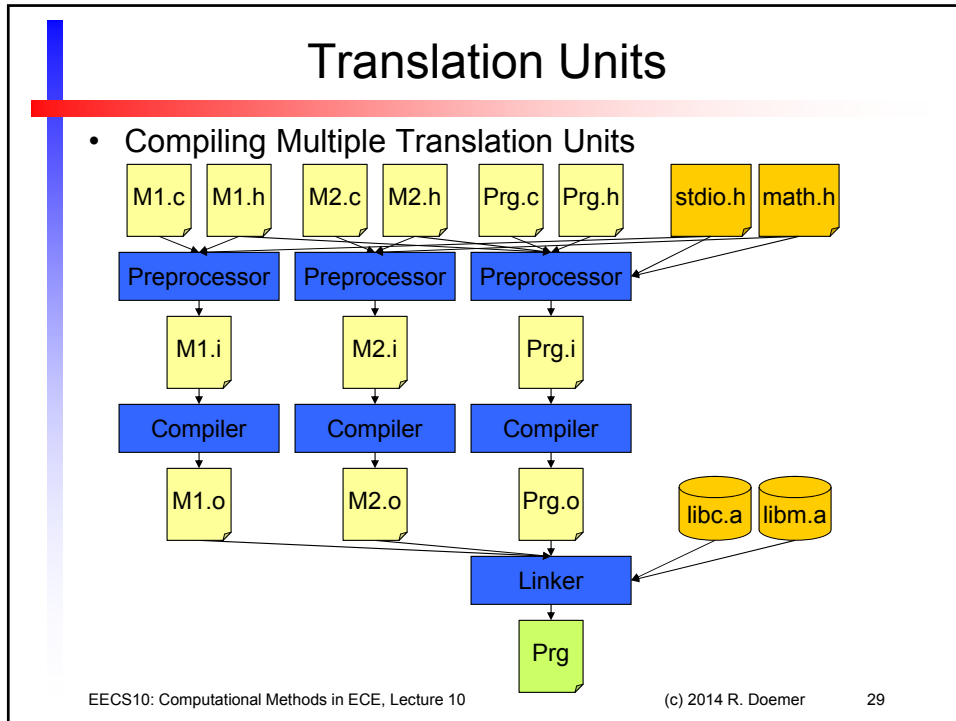
## Translation Units

- Multiple Translation Units
  - C programs can be partitioned into multiple translation units, aka. *modules*
  - Modules typically consist of
    - Module header file (file suffix `.h`)
    - Module program file (file suffix `.c`)
    - Module object file (file suffix `.o`)
  - Modules are *linked* together
    - Linker combines object files and required libraries into an executable file
    - `gcc Program.o Mod1.o Mod2.o -lc -lm -Wall -ansi -o Program`

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

28



## Translation Units

- Example: Header file `Constants.h`

```

/*****
/* Constants.h: header file for constant definitions */
/* author: Rainer Doemer */
/* modifications: (most recent first) */
/* 07/22/14 RD version for Summer 2014 */
*****/

#ifndef CONSTANTS_H
#define CONSTANTS_H

/** global definitions */

#define WIDTH 810 /* image width */
#define HEIGHT 450 /* image height */
#define SLEN 80 /* max. string length */

#endif /* CONSTANTS_H */

/* EOF Constants.h */

```

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

31

## Translation Units

- Example: Header file `FileIO.h`

```

/*****
/* FileIO.h: header file for I/O module */
*****/
#ifndef FILE_IO_H
#define FILE_IO_H

#include "Constants.h"

int ReadImage( /* read image from file */
    char Filename[SLEN],
    unsigned char R[WIDTH][HEIGHT],
    unsigned char G[WIDTH][HEIGHT],
    unsigned char B[WIDTH][HEIGHT]);

int WriteImage( /* write image to file */
    char Filename[SLEN],
    unsigned char R[WIDTH][HEIGHT],
    unsigned char G[WIDTH][HEIGHT],
    unsigned char B[WIDTH][HEIGHT]);

#endif /* FILE_IO_H */
/* EOF FileIO.h */

```

EECS



## Translation Units

- Example: Program file `FileIO.c`

```

/*****
/* FileIO.c: program file for I/O module          */
/*****

#include <stdio.h>
#include "FileIO.h"

/**/ function definitions ***/

int ReadImage(char Filename[SLEN],
              unsigned char R[WIDTH][HEIGHT],
              unsigned char G[WIDTH][HEIGHT],
              unsigned char B[WIDTH][HEIGHT])
{ /* ... function body ... */
} /* end of ReadImage */

int WriteImage(char Filename[SLEN],
               unsigned char R[WIDTH][HEIGHT],
               unsigned char G[WIDTH][HEIGHT],
               unsigned char B[WIDTH][HEIGHT])
{ /* ... function body ... */
} /* end of WriteImage */
EECS /* EOF FileIO.c */

```

## Translation Units

- Example: Header file `Age.h`

```

/*****
/* Age.h: header file for aging operation        */
/*****

#ifndef AGE_H
#define AGE_H

/**/ header files ***/

#include "Constants.h"

/**/ function declarations ***/

void Age( /* age the image */
         unsigned char R[WIDTH][HEIGHT],
         unsigned char G[WIDTH][HEIGHT],
         unsigned char B[WIDTH][HEIGHT]);

#endif /* AGE_H */

/* EOF Age.h */

```

## Translation Units

- Example: Program file `Age.c`

```

/*****
/* Age.c: program file for aging operation */
/*****

#include "Age.h"

/** function definitions */

/* age the image so that it looks like an old photo */

void Age(
    unsigned char R[WIDTH][HEIGHT],
    unsigned char G[WIDTH][HEIGHT],
    unsigned char B[WIDTH][HEIGHT])
{
    /* ... function body ... */
} /* end of Age */

/* EOF Age.c */

```

## Translation Units

- Example: Program file `Main.c`

```

/*****
/* Main.c: main program file */
/*****

#include "Constants.h"
#include "FileIO.h"
#include "Age.h"

int main(void)
{
    unsigned char R[WIDTH][HEIGHT];
    unsigned char G[WIDTH][HEIGHT];
    unsigned char B[WIDTH][HEIGHT];

    if (ReadImage("UCI_Firetrucks.ppm", R, G, B) != 0)
    { return 10; }
    Age(R, G, B);
    if (WriteImage("UCI_Firetrucks_old.ppm", R, G, B) != 0)
    { return 10; }

    return 0;
} /* end of main */

/* EOF Main.c */

```

## Translation Units

- Example session:

```
% vi Constants.h
% vi FileIO.h
% vi FileIO.c
% vi Age.h
% vi Age.c
% vi Main.c
```

```
% gcc -c FileIO.c -o FileIO.o -Wall -ansi
% gcc -c Age.c -o Age.o -Wall -ansi
% gcc -c Main.c -o Main.o -Wall -ansi
% gcc FileIO.o Age.o Main.o -o PhotoLab2
% PhotoLab2
%
```

WorldCup2014.ppm



WorldCup1954.ppm



EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

37

## Lecture 10.3: Overview

- Review
  - Lecture 8.1: Recursion
  - Lecture 8.2: Structures, unions, enumerators
  - Lecture 9.1: Binary data representation, memory
  - Lecture 9.2: Pointers, pointer operations
  - Lecture 9.3: String operations using pointers
  - Lecture 10.1: File processing
  - Lecture 10.2: Translation units
- Review Quiz

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

38

## Midterm 2 Review Quiz

- Top 5 most “difficult” questions:
  - In the program below, what is the result of calling `grade(75)`?


- a) 'A'
- b) 'B'
- c) 'C'
- d) 'D'
- e) 'F'

```

1 char grade(int x)
2 { char g;
3   if (x > 90)
4     { g = 'A'; }
5   if (x > 80)
6     { g = 'B'; }
7   if (x > 70)
8     { g = 'C'; }
9   if (x > 60)
10    { g = 'D'; }
11  else
12    { g = 'F'; }
13  return g;
14 }
```

## Midterm 2 Review Quiz

- Top 5 most “difficult” questions:
  - In the program below, what is the result of calling `grade(75)`?

- a) 'A'
- b) 'B'
- c) 'C'
-  d) 'D'
- e) 'F'

```

1 char grade(int x)
2 { char g;
3   if (x > 90)
4     { g = 'A'; }
5   if (x > 80)
6     { g = 'B'; }
7   if (x > 70)
8     { g = 'C'; }
9   if (x > 60)
10    { g = 'D'; }
11  else
12    { g = 'F'; }
13  return g;
14 }
```

## Midterm 2 Review Quiz

- Top 5 most “difficult” questions:
  - In the program below, what is the result of calling `grade(80-90)`?

- a) 'A'
- b) 'B'
- c) 'C'
- d) 'D'
- e) 'F'

```

1 char grade(int x)
2 { char g;
3   if (x > 90)
4     { g = 'A'; }
5   if (x > 80)
6     { g = 'B'; }
7   if (x > 70)
8     { g = 'C'; }
9   if (x > 60)
10    { g = 'D'; }
11  else
12    { g = 'F'; }
13  return g;
14 }
```

## Midterm 2 Review Quiz

- Top 5 most “difficult” questions:
  - In the program below, what is the result of calling `grade(80-90)`?

- a) 'A'
- b) 'B'
- c) 'C'
- d) 'D'
-  e) 'F'

```

1 char grade(int x)
2 { char g;
3   if (x > 90)
4     { g = 'A'; }
5   if (x > 80)
6     { g = 'B'; }
7   if (x > 70)
8     { g = 'C'; }
9   if (x > 60)
10    { g = 'D'; }
11  else
12    { g = 'F'; }
13  return g;
14 }
```

## Midterm 2 Review Quiz

- Top 5 most “difficult” questions:
    - Which of the following are valid definitions of an integer array **A** of size 3?  
(Check all that apply!)
- a) `int A[3];`  
 b) `int A[3] = {1,2,3};`  
 c) `int A[3] = {};`  
 d) `int A[3] = {1, 2};`  
 e) `int A[ ] = {1,2,3};`






EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

43

## Midterm 2 Review Quiz

- Top 5 most “difficult” questions:
  - Which of the following are valid definitions of an integer array **A** of size 3?  
(Check all that apply!)

-  a) `int A[3];`  
 b) `int A[3] = {1,2,3};`  
 c) `int A[3] = {};`  
 d) `int A[3] = {1, 2};`  
 e) `int A[ ] = {1,2,3};`

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

44

## Midterm 2 Review Quiz

- Top 5 most “difficult” questions:
  - Given two global variables `int x=7` and `int y=8`, which of the following functions properly swaps the values such that `x=8` and `y=7`? (Check all that apply!)

a) 

```
void swap(int x, int y)
{ x = y; y = x;
}
```

b) 

```
void swap(void)
{ x = y; y = x;
}
```

c) 

```
void swap(void)
{ int t;
  t = x; x = y; y = t;
}
```

d) 

```
void swap(void)
{ int t;
  t = y; y = x; x = t;
}
```

e) 

```
void swap(int x, int y)
{ int t;
  t = x; x = y; y = t;
}
```

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

45

## Midterm 2 Review Quiz

- Top 5 most “difficult” questions:
  - Given two global variables `int x=7` and `int y=8`, which of the following functions properly swaps the values such that `x=8` and `y=7`? (Check all that apply!)

a) 

```
void swap(int x, int y)
{ x = y; y = x;
}
```

b) 

```
void swap(void)
{ x = y; y = x;
}
```

c) 

```
void swap(void)
{ int t;
  t = x; x = y; y = t;
}
```

d) 

```
void swap(void)
{ int t;
  t = y; y = x; x = t;
}
```

e) 

```
void swap(int x, int y)
{ int t;
  t = x; x = y; y = t;
}
```

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

46

## Midterm 2 Review Quiz

- Top 5 most “difficult” questions:
    - In the `gdb` debugger, which commands allow you to run your program step by step? (Check all that apply!)
- a) `step`
  - b) `cont`
  - c) `run`
  - d) `next`
  - e) `back`

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

47

## Midterm 2 Review Quiz

- Top 5 most “difficult” questions:
    - In the `gdb` debugger, which commands allow you to run your program step by step? (Check all that apply!)
- a) `step`
  - b) `cont`
  - c) `run`
  - d) `next`
  - e) `back`

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

48



## Quiz: Question 1

- In the program below, what is printed by the function call `g(1)`?

- a) 1 2
- b) 2 3
- c) 1 1
- d) 2
- e) 1


```

1 int f(int x)
2 { printf("%d ", x);
3   return x + 1;
4 }
5 int g(int x)
6 { printf("%d ", f(x));
7   return x + 2;
8 }

```

## Quiz: Question 1

- In the program below, what is printed by the function call `g(1)`?

-  a) 1 2
- b) 2 3
- c) 1 1
- d) 2
- e) 1

```

1 int f(int x)
2 { printf("%d ", x);
3   return x + 1;
4 }
5 int g(int x)
6 { printf("%d ", f(x));
7   return x + 2;
8 }

```

## Quiz: Question 2

- What is recursion?  
(Check all that apply!)
  - a) A function that does not terminate.
  - b) A function that calls itself.
  - c) A function that contains a loop.
  - d) A function  $f$  that calls a function  $g$  which calls  $f$ .
  - e) A function that returns no value.

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

51

## Quiz: Question 2

- What is recursion?  
(Check all that apply!)
  - a) A function that does not terminate.
  - b) A function that calls itself.
  - c) A function that contains a loop.
  - d) A function  $f$  that calls a function  $g$  which calls  $f$ .
  - e) A function that returns no value.

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

52

### Quiz: Question 3

- Given the function definition below, what is printed for the function call  $f(3)$ ?


- a) 1 2 3
- b) 1 2 3 4
- c) 3 2 1 0
- d) 4 3 2 1
- e) 3 2 1

```

1 void f(int x)
2 {
3     printf("%d ", x);
4     if (x > 0)
5         { f(x-1); }
6 }
```

### Quiz: Question 3

- Given the function definition below, what is printed for the function call  $f(3)$ ?

- a) 1 2 3
- b) 1 2 3 4
-  c) 3 2 1 0
- d) 4 3 2 1
- e) 3 2 1

```

1 void f(int x)
2 {
3     printf("%d ", x);
4     if (x > 0)
5         { f(x-1); }
6 }
```

## Quiz: Question 4

- Given the following definition of the vectors  $\mathbf{v}_1$ ,  $\mathbf{v}_2$  and  $\mathbf{v}_3$ , what is a correct way to perform a vector addition of  $\mathbf{v}_1$  and  $\mathbf{v}_2$ ?

```
struct v {int x, y;} v1, v2, v3;
```

- $\mathbf{v}_3 = \mathbf{v}_1 + \mathbf{v}_2;$
- $\mathbf{v}_3 = \mathbf{v}_1[\mathbf{x}] * \mathbf{v}_2[\mathbf{y}] + \mathbf{v}_1[\mathbf{y}] * \mathbf{v}_2[\mathbf{x}]$
- $\mathbf{v}_3[0] = \mathbf{v}_1[0] + \mathbf{v}_2[0];$   
 $\mathbf{v}_3[1] = \mathbf{v}_1[1] + \mathbf{v}_2[1];$
- $\mathbf{v}_3.\mathbf{x} = \mathbf{v}_1.\mathbf{x} + \mathbf{v}_2.\mathbf{x};$   
 $\mathbf{v}_3.\mathbf{y} = \mathbf{v}_1.\mathbf{y} + \mathbf{v}_2.\mathbf{y};$
- $\mathbf{v}_3 \rightarrow \mathbf{x} = \mathbf{v}_1 \rightarrow \mathbf{x} + \mathbf{v}_2 \rightarrow \mathbf{x};$   
 $\mathbf{v}_3 \rightarrow \mathbf{y} = \mathbf{v}_1 \rightarrow \mathbf{y} + \mathbf{v}_2 \rightarrow \mathbf{y};$

EECS10: Computational Methods in ECE, Lecture 10


(c) 2014 R. Doemer

55

## Quiz: Question 4

- Given the following definition of the vectors  $\mathbf{v}_1$ ,  $\mathbf{v}_2$  and  $\mathbf{v}_3$ , what is a correct way to perform a vector addition of  $\mathbf{v}_1$  and  $\mathbf{v}_2$ ?

```
struct v {int x, y;} v1, v2, v3;
```

- $\mathbf{v}_3 = \mathbf{v}_1 + \mathbf{v}_2;$
- $\mathbf{v}_3 = \mathbf{v}_1[\mathbf{x}] * \mathbf{v}_2[\mathbf{y}] + \mathbf{v}_1[\mathbf{y}] * \mathbf{v}_2[\mathbf{x}]$
- $\mathbf{v}_3[0] = \mathbf{v}_1[0] + \mathbf{v}_2[0];$   
 $\mathbf{v}_3[1] = \mathbf{v}_1[1] + \mathbf{v}_2[1];$
-   $\mathbf{v}_3.\mathbf{x} = \mathbf{v}_1.\mathbf{x} + \mathbf{v}_2.\mathbf{x};$   
 $\mathbf{v}_3.\mathbf{y} = \mathbf{v}_1.\mathbf{y} + \mathbf{v}_2.\mathbf{y};$
- $\mathbf{v}_3 \rightarrow \mathbf{x} = \mathbf{v}_1 \rightarrow \mathbf{x} + \mathbf{v}_2 \rightarrow \mathbf{x};$   
 $\mathbf{v}_3 \rightarrow \mathbf{y} = \mathbf{v}_1 \rightarrow \mathbf{y} + \mathbf{v}_2 \rightarrow \mathbf{y};$

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

56

## Quiz: Question 5

- Given the following enumerator definition, what is printed by `printf("%d", two);`?


```
enum count {one, two, three, four = 4};
```

- a) one
- b) two
- c) three
- d) 1
- e) 2

## Quiz: Question 5

- Given the following enumerator definition, what is printed by `printf("%d", two);`?

```
enum count {one, two, three, four = 4};
```

- a) one
- b) two
- c) three
-  d) 1
- e) 2

## Quiz: Question 6

- Which of the following components do you find in every computer?  
(Check all that apply!)
- a) **ROM**
  - b) **RUM**
  - c) **BUG**
  - d) **CPU**
  - e) **IBM**

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

59

## Quiz: Question 6

- Which of the following components do you find in every computer?  
(Check all that apply!)
- a) **ROM**
  - b) **RUM**
  - c) **BUG**
  - d) **CPU**
  - e) **IBM**

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

60

### Quiz: Question 7


- What is the decimal value of the (unsigned) binary number  $01010101_2$  ?  
  
a) 01010101  
b) 85  
c) 101  
d) 170  
e) 255

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

61

### Quiz: Question 7

- What is the decimal value of the (unsigned) binary number  $01010101_2$  ?  
  
a) 01010101  
 b) 85  
c) 101  
d) 170  
e) 255

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

62

### Quiz: Question 8

- What is the binary value of the hexadecimal number  $FF_{16}$  ?  
  
a) 01010101  
b) 10001000  
c) 01110111  
d) 00010001  
e) 11111111

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

63

### Quiz: Question 8

- What is the binary value of the hexadecimal number  $FF_{16}$  ?  
  
a) 01010101  
b) 10001000  
c) 01110111  
d) 00010001  
e) 11111111

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

64



### Quiz: Question 9


- How many bits do you need to represent one hexadecimal digit?
- a) 1  
b) 2  
c) 4  
d) 8  
e) 16

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

65

### Quiz: Question 9

- How many bits do you need to represent one hexadecimal digit?
- a) 1  
b) 2  
 c) 4  
d) 8  
e) 16

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

66

## Quiz: Question 10

- What could cause a **bus error**?  
(Check all that apply!)
  - a) Waking up late and missing the bus.
  - b) Calling a recursive function.
  - c) Accessing an array with an index out of range.
  - d) Referencing a pointer variable with invalid value.
  - e) Accessing an integer variable with invalid value.

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

67

## Quiz: Question 10

- What could cause a **bus error**?  
(Check all that apply!)
  - a) Waking up late and missing the bus.
  - b) Calling a recursive function.
  - c) Accessing an array with an index out of range.
  - d) Referencing a pointer variable with invalid value.
  - e) Accessing an integer variable with invalid value.

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

68

## Quiz: Question 11

- In C, which properties does every object have?  
(Check all that apply!)
  - a) A size.
  - b) A value.
  - c) A weight.
  - d) A type.
  - e) A location.

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

69

## Quiz: Question 11

- In C, which properties does every object have?  
(Check all that apply!)
  - a) A size.
  - b) A value.
  - c) A weight.
  - d) A type.
  - e) A location.

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

70

## Quiz: Question 12

- Given the program segment below, what is the value of `*p` at the end?

- a) 1
- b) 2
- c) 3
- d) 4
- e) 5

```

1 int x[] = {1,2,3,4,5};
2 int *p = &x[2];
3
4 p++;
5 p -= 2;

```

## Quiz: Question 12

- Given the program segment below, what is the value of `*p` at the end?

- a) 1
-  b) 2
- c) 3
- d) 4
- e) 5

```

1 int x[] = {1,2,3,4,5};
2 int *p = &x[2];
3
4 p++;
5 p -= 2;

```

## Quiz: Question 13

- Given the function and variable definitions shown below, which function call is valid? (Check all that apply!)

- a) `StrLen(cp);`
- b) `StrLen(ca);`
- c) `StrLen(c);`
- d) `StrLen(i);`
- e) `StrLen("abc");`

```

1 int StrLen(
2     const char *s)
3 { int l = 0;
4   while(*s)
5     { s++;
6       l++;
7     }
8   return l;
9 }
10 char *cp = "hello";
11 char ca[] = "world";
12 char c = 'c';
13 int i = 42;

```

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

73

## Quiz: Question 13

- Given the function and variable definitions shown below, which function call is valid? (Check all that apply!)

- a) `StrLen(cp);`
- b) `StrLen(ca);`
- c) `StrLen(c);`
- d) `StrLen(i);`
- e) `StrLen("abc");`

```

1 int StrLen(
2     const char *s)
3 { int l = 0;
4   while(*s)
5     { s++;
6       l++;
7     }
8   return l;
9 }
10 char *cp = "hello";
11 char ca[] = "world";
12 char c = 'c';
13 int i = 42;

```

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

74

## Quiz: Question 14

- Which of the following are functions declared in `stdio.h`?  
(Check all that apply!)
  - a) `printf`
  - b) `printfd`
  - c) `fprintf`
  - d) `sprint`
  - e) `fputs`

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

75

## Quiz: Question 14

- Which of the following are functions declared in `stdio.h`?  
(Check all that apply!)
  - a) `printf`
  - b) `printfd`
  - c) `fprintf`
  - d) `sprint`
  - e) `fputs`

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

76

## Quiz: Question 15

- What does the following code segment print?

```

1 char s[] = "Hppe!mvd1!boe!fokpz!uif!tvnnfs";
2 char *p;
3 p = &s[0];
4 while(*p)
5 { printf("%c", *p - 1);
6   p++;
7 }

```

- Hppe!mvd1!boe!fokpz!uif!tvnnfs
- Have fun and enjoy the summer
- Enjoy the summer and have fun
- Good summer and enjoy the luck
- Good luck and enjoy the summer

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

77


## Quiz: Question 15

- What does the following code segment print?

```

1 char s[] = "Hppe!mvd1!boe!fokpz!uif!tvnnfs";
2 char *p;
3 p = &s[0];
4 while(*p)
5 { printf("%c", *p - 1);
6   p++;
7 }

```

- Hppe!mvd1!boe!fokpz!uif!tvnnfs
- Have fun and enjoy the summer
- Enjoy the summer and have fun
- Good summer and enjoy the luck
-  Good luck and enjoy the summer

EECS10: Computational Methods in ECE, Lecture 10

(c) 2014 R. Doemer

78