

EECS 10: Computational Methods in Electrical and Computer Engineering

Lecture 18

Rainer Dömer

doemer@uci.edu

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

Lecture 18: Overview

- Midterm Course Evaluation
 - Results
- Recursion
 - Introduction
 - Concept of recursion
 - Recursion vs. iteration
 - Examples
 - Factorial function: `Factorial.c`
 - Fibonacci series: `Fibonacci.c`

Midterm Course Evaluation: Results

- Only few respondents
 - 45 out of 175 (<26%) for lectures, even less for labs/discussions
- Overall feedback
 - Many positive comments (“great”, “effective”, “funny”, “knowledgable”)
 - Few negative comments (“not helpful”, “not fair”?)
- Suggested improvements
 - Lectures
 - “Slow down”
 - “more hands-on experience”
 - “more specific on readings in Deitel text” / “Matches up with the book well”
 - “more example programs”
 - “I would like to see how the examples are run in class”
 - “more warm up quizzes”
 - Homework
 - “go over the homework in class”
 - “more helpful hints towards finishing the homework”
 - “more on how to do the homework assignments”

EECS10: Computational Methods in ECE, Lecture 18

(c) 2006 R. Doemer

3

Recursion

- Introduction
 - *Recursion* is often an alternative to *Iteration*
 - Recursion is a very simple concept, yet very powerful
 - Recursion is present in nature
 - Trees have branches, which have branches, which have branches, ... which have leaves.
 - Recursion is traversal of hierarchy
 - *Traverse* (climb) a tree to the top:
 - start at the root
 - at a leaf, stop
 - at a branch, *traverse* one branch
 - *Traverse* a file system on a computer
 - start at the current directory
 - at a file, process the file
 - at a directory, *traverse* the directory

EECS10: Computational Methods in ECE, Lecture 18

(c) 2006 R. Doemer

4

Recursion

- **Recursive Function**
 - Function that calls itself ...
 - ... directly, or
 - ... indirectly
- **Concept of Recursion**
 - Trivial *base case*
 - Return value defined for simple case
 - Example: `if (arg == 0) {return 1; }`
 - *Recursion step*
 - Reduce the problem towards the base case
 - Make a recursive function call
 - Example: `if (arg > 0) { return ...fct(arg-1); }`
- **Termination of Recursion**
 - Converging of recursive calls to the base case
 - Recursive call must be “simpler” than current call

```
int f(...)
{ ...
  f(...);
  ...
}
```

```
int a(...)
{ ...
  b(...);
  ...
}
int b(...)
{ ...
  a(...);
  ...
}
```

EECS10: Computational Methods in ECE, Lecture 18

(c) 2006 R. Doemer

5

Recursion

- **Example: Factorial function $n!$**
 - The factorial of a non-negative integer is
 - $n! = n * (n-1) * (n-2) * (n-3) * \dots * 1$
 - This can be written as
 - $n! = n * ((n-1) * ((n-2) * ((n-3) * (\dots * 1))))$
 - *Recursive definition:*
 - $n=1: 1! = 1$ (base case)
 - $n>1: n! = n * (n-1)!$ (recursion step)
 - Example computation:

```
5! = 5 * 4!
   = 5 * (4 * 3!)
   = 5 * (4 * (3 * 2!))
   = 5 * (4 * (3 * (2 * 1!)))
   = 5 * (4 * (3 * (2 * 1)))
   = 5 * (4 * (3 * 2))
   = 5 * (4 * 6)
   = 5 * 24
   = 120
```

EECS10: Computational Methods in ECE, Lecture 18

(c) 2006 R. Doemer

6

Recursion

- Program example: `Factorial.c` (part 1/2)

```

/* Factorial.c: example demonstrating recursion */
/* author: Rainer Doemer */
/* modifications: */
/* 11/14/04 RD initial version */

#include <stdio.h>

/* function definition */
long factorial(long n)
{
    if (n == 1) /* base case */
    { return 1;
      } /* fi */
    else /* recursion step */
    { return n * factorial(n-1);
      } /* esle */
} /* end of factorial */

...

```

EECS10: Computational Methods in ECE, Lecture 18

(c) 2006 R. Doemer

7

Recursion

- Program example: `Factorial.c` (part 2/2)

```

...
int main(void)
{
    /* variable definitions */
    long int n, f;

    /* input section */
    printf("Please enter value n: ");
    scanf("%ld", &n);

    /* computation section */
    f = factorial(n);

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

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

/* EOF */

```

EECS10: Computational Methods in ECE, Lecture 18

(c) 2006 R. Doemer

8

Recursion

- Example session: **Factorial.c**

```
% vi Factorial.c
% gcc Factorial.c -o Factorial -Wall -ansi
% Factorial
Please enter value n: 1
The factorial of 1 is 1.
% Factorial
Please enter value n: 2
The factorial of 2 is 2.
% Factorial
Please enter value n: 3
The factorial of 3 is 6.
% Factorial
Please enter value n: 5
The factorial of 5 is 120.
% Factorial
Please enter value n: 10
The factorial of 10 is 3628800.
%
```

EECS10: Computational Methods in ECE, Lecture 18

(c) 2006 R. Doemer

9

Recursion vs. Iteration

- Example: Factorial function $n!$

- The factorial of a non-negative integer is

- $n! = n * (n-1) * (n-2) * (n-3) * \dots * 1$

- This can be written as

- $n! = n * ((n-1) * ((n-2) * ((n-3) * (\dots * 1))))$

- *Recursive* definition:

- $n=1: 1! = 1$ (base case)

- $n>1: n! = n * (n-1)!$ (recursion step)

- *Iterative* implementation:

- Compute n products in a loop

- $n! = n * (n-1) * (n-2) * (n-3) * \dots * 1$

- $p = n;$

- for ($f=n-1; f>=1; f--$)

- { $p = p * f;$ }

EECS10: Computational Methods in ECE, Lecture 18

(c) 2006 R. Doemer

10

Recursion vs. Iteration

- Program example: `Factorial2.c` (part 1/2)

```

/* Factorial2.c: example demonstrating iteration */
/* author: Rainer Doemer */
/* modifications: */
/* 11/14/04 RD initial version (based on Factorial.c) */
#include <stdio.h>

/* function definition */
long factorial(long n)
{
    long product, factor;

    product = n;
    for(factor = n-1; factor >=1; factor--)
        { product *= factor;
          } /* rof */
    return product;
} /* end of factorial */

...

```

EECS10: Computational Methods in ECE, Lecture 18

(c) 2006 R. Doemer

11

Recursion vs. Iteration

- Program example: `Factorial2.c` (part 2/2)

```

...
int main(void)
{
    /* variable definitions */
    long int n, f;

    /* input section */
    printf("Please enter value n: ");
    scanf("%ld", &n);

    /* computation section */
    f = factorial(n);

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

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

/* EOF */

```

EECS10: Computational Methods in ECE, Lecture 18

(c) 2006 R. Doemer

12

Recursion vs. Iteration

- Example session: **Factorial2.c**

```
% cp Factorial.c Factorial2.c
% vi Factorial2.c
% gcc Factorial2.c -o Factorial2 -Wall -ansi
% Factorial2
Please enter value n: 1
The factorial of 1 is 1.
% Factorial2
Please enter value n: 2
The factorial of 2 is 2.
% Factorial2
Please enter value n: 3
The factorial of 3 is 6.
% Factorial2
Please enter value n: 5
The factorial of 5 is 120.
% Factorial2
Please enter value n: 10
The factorial of 10 is 3628800.
%
```

EECS10: Computational Methods in ECE, Lecture 18

(c) 2006 R. Doemer

13

Recursion

- Example 2: Fibonacci series

- Sequence of integers
 - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, ...
- Mathematical properties
 - The first two numbers are 0 and 1
 - Every subsequent Fibonacci number is the sum of the previous two Fibonacci numbers
- Ratio of successive Fibonacci numbers is ...
 - ... converging to constant value 1.618...
 - ... called *Golden Ratio* or *Golden Mean*
- Recursive definition:
 - Base case: $fibonacci(0) = 0$
 $fibonacci(1) = 1$
 - Recursion step: $fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)$

EECS10: Computational Methods in ECE, Lecture 18

(c) 2006 R. Doemer

14

Recursion

- Program example: `Fibonacci.c` (part 1/2)

```

/* Fibonacci.c: example demonstrating recursion */
/* author: Rainer Doemer */
/* modifications: */
/* 11/14/04 RD initial version */

#include <stdio.h>

/* function definition */
long fibonacci(long n)
{
    if (n <= 1) /* base case */
    { return n;
      } /* fi */
    else /* recursion step */
    { return fibonacci(n-1) + fibonacci(n-2);
      } /* esle */
} /* end of fibonacci */

/* main function */
...

```

EECS10: Computational Methods in ECE, Lecture 18

(c) 2006 R. Doemer

15

Recursion

- Program example: `Fibonacci.c` (part 2/2)

```

...
int main(void)
{
    /* variable definitions */
    long int n, f;

    /* input section */
    printf("Please enter value n: ");
    scanf("%ld", &n);

    /* computation section */
    f = fibonacci(n);

    /* output section */
    printf("The %ld-th Fibonacci number is %ld.\n", n, f);

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

/* EOF */

```

EECS10: Computational Methods in ECE, Lecture 18

(c) 2006 R. Doemer

16

Recursion

- Example session: **Fibonacci.c**

```
% cp Factorial.c Fibonacci.c
% vi Fibonacci.c
% gcc Fibonacci.c -o Fibonacci -Wall -ansi
% Fibonacci
Please enter value n: 1
The 1-th Fibonacci number is 1.
% Fibonacci
Please enter value n: 10
The 10-th Fibonacci number is 55.
% Fibonacci
Please enter value n: 20
The 20-th Fibonacci number is 6765.
% Fibonacci
Please enter value n: 30
The 30-th Fibonacci number is 832040.
% Fibonacci
Please enter value n: 40
The 40-th Fibonacci number is 102334155.
%
```