EECS 10: Computational Methods in Electrical and Computer Engineering

Lecture 4

Rainer Dömer

doemer@uci.edu

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

Lecture 4: Overview

- Warm-up Quiz
- Our first C Program
  - Example **`HelloWorld.c`**
  - Character strings and escape sequences
- Our second C Program
  - Example **`Addition.c`**
  - Program structure
  - Variables
  - Data input
  - Computation
  - Data output

Quiz: Question 4

- What is C *not*?
  a) a structured programming language
  b) a compiled programming language
  c) a high-level programming language
  d) a portable programming language
  e) a object-oriented programming language

Quiz: Question 4

- What is C *not*?
  a) a structured programming language
  b) a compiled programming language
  c) a high-level programming language
  d) a portable programming language
  e) a object-oriented programming language

Quiz: Question 5

- What is the meaning of the following code fragment?

```
/* printf("C programming is great!\n") */
```

a) it prints “C programming is boring!”
b) it is the main function of the C program
c) it is a comment ignored by the compiler
d) it prints “C programming is great!”
e) it is a syntax error because a semicolon is missing after the **`printf()`** statement

Quiz: Question 5

- What is the meaning of the following code fragment?

```
/* printf("C programming is great!\n") */
```

a) it prints “C programming is boring!”
b) it is the main function of the C program
c) it is a comment ignored by the compiler
d) it prints “C programming is great!”
e) it is a syntax error because a semicolon is missing after the **`printf()`** statement

Quiz: Question 6

- What is *not* true about of the following compiler call?

```
% gcc –Wall –ansi HelloWorld.c –o HelloWorld
```

a) the GNU C Compiler is called to generate an executable program called **`HelloWorld`**

b) the compiler will print warning and/or error messages about any non-ANSI compliance in the code

c) the compiler will read the file **`HelloWorld.c`**

d) the compiler will ignore all warnings

e) the compiler will assume that **`HelloWorld.c`** is an ANSI-compliant C program

Quiz: Question 6

- What is *not* true about of the following compiler call?

```
% gcc –Wall –ansi HelloWorld.c –o HelloWorld
```

a) the GNU C Compiler is called to generate an executable program called **`HelloWorld`**

b) the compiler will print warning and/or error messages about any non-ANSI compliance in the code

c) the compiler will read the file **`HelloWorld.c`**

d) the compiler will ignore all warnings

e) the compiler will assume that **`HelloWorld.c`** is an ANSI-compliant C program

Our first C Program

- Program example: H**elloWorld.c**

```
/* HelloWorld.c: our first C program    */
/*                                      */
/* author: Rainer Doemer                */
/*                                      */
/* modifications:                       */
/* 09/28/04 RD  initial version         */

#include <stdio.h>

/* main function */

int main(void)
{
    printf("Hello World!\n");
    return 0;
}

/* EOF */
```


Our first C Program

- Character string constants: “Strings”
  - start and end with a double quote character (**"**)
  - may not extend over a single line
  - subsequent string constants are combined
  - text formatting using escape sequences
    - **\n** new line
    - **\t** horizontal tab
    - **\r** carriage return
    - **\b** back space
    - **\a** alert / bell
    - **** backslash character
    - **\"** double quote character
- Experiments with the **HelloWorld** program...

Our second C Program

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

```
/* Addition.c: adding two integer numbers       */
/*                                              */
/* author: Rainer Doemer                        */
/*                                              */
/* modifications:                               */
/* 09/30/04 RD  initial version                 */

#include <stdio.h>

/* main function */

int main(void)
{
    /* variable definitions */
    int i1 = 0;       /* first integer */
    int i2 = 0;       /* second integer */
    int sum;          /* result */
...
```


Our second C Program

- Program example: Addition.c (part 2/2)

```
...
    /* input section */
    printf("Please enter an integer:      ");
    scanf("%d", &i1);
    printf("Please enter another integer: ");
    scanf("%d", &i2);

    /* computation section */
    sum = i1 + i2;

    /* output section */
    printf("The sum of %d and %d is %d.\n", i1, i2, sum);

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

/* EOF */
```

Our second C Program

- Program structure
  - Variable definition and initialization
    - define and name the storage elements needed
    - define the type of the storage elements
    - define the initial values of the storage elements
  - Input section
    - read the input values needed for the computation
  - Computation section
    - perform the necessary computation
  - Output section
    - output the results of the computation
  - Exit section
    - clean up and exit

Our second C Program

- Variable definition and initialization

```
/* variable definitions */
int i1 = 0;       /* first integer */
int i2 = 0;       /* second integer */
int sum;          /* result */
```

  - Variable type: **`int`**
    - integer type, stores whole numbers (e.g. -5, 0, 42)
    - many other types exist (**`float`**, **`double`**, **`char`**, ...)
  - Variable name: **`i1`**, **`i2`**, **`sum`**
    - valid identifier, i.e. name composed of letters, digits
    - variable name should be descriptive
  - Initializer: **`= 0`**
    - optional (if left out, initial value is undefined)
    - specifies the initial value of the variable

Our second C Program

- Data input using **`scanf()`** function

```
/* input section */
printf("Please enter an integer:      ");
scanf("%d", &i1);
```

  - part of standard I/O library
    - declared in header file **`stdio.h`**
  - reads data from the standard input stream **`stdin`**
    - **`stdin`** usually means the keyboard
  - converts input data according to format string
    - **`"%d"`** indicates that a decimal integer value is expected
  - stores result in specified location
    - **`&i1`** indicates to store at the *address of* variable **`i1`**

Our second C Program

- Computation using assignment statements

```
/* computation section */
sum = i1 + i2;
```

  - Operator **=** specifies an assignment
    - value of the right-hand side (**`i1 + i2`**) is assigned to the left-hand side (**`sum`**)
    - left-hand side is usually a variable
    - right-hand side is a simple or complex expression
  - Operator **+** specifies addition
    - left and right arguments are added
    - result is the sum of the two arguments
  - May other operators exist
    - For example, **-**, *****, **/**, **%**, **<**, **>**, **==**, **^**, **&**, **|**, ...

Our second C Program

- Data output using **`printf()`** function

```
/* output section */
printf("The sum of %d and %d is %d.\n", i1, i2, sum);
```

  - part of standard I/O library
    - declared in header file **`stdio.h`**
  - writes data to the standard output stream **`stdout`**
    - **`stdout`** usually means the monitor
  - converts output data according to format string
    - standard text is copied verbatim to the output
    - **`"%d"`** is replaced with a decimal integer value
  - takes values from specified arguments
    - **`i1`** indicates to use the value of the variable **`i1`**

Our second C Program

- Example session: Addition.c

```
% vi Addition.c
% ls -l
-rw-------   1 doemer   faculty      702 Sep 30 14:17 Addition.c
% gcc -Wall -ansi Addition.c -o Addition
% ls -l
-rwx------   1 doemer   faculty     6628 Sep 30 16:44 Addition*
-rw-------   1 doemer   faculty      702 Sep 30 14:17 Addition.c
% Addition
Please enter an integer:      27
Please enter another integer: 15
The sum of 27 and 15 is 42.
% Addition
Please enter an integer:      123
Please enter another integer: -456
The sum of 123 and -456 is -333.
%
```