

EECS 10: Computational Methods in Electrical and Computer Engineering

Lecture 3

Rainer Dömer

doemer@uci.edu

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

Lecture 3: Overview

- Introduction to Programming in C
 - History of C
 - Introduction to C
- Our first C Program
 - Example `HelloWorld.c`
 - Structure of a C program
 - `printf()` function
 - Program compilation and execution
 - String constants

Introduction to Programming

- Categories of programming languages
 - Machine languages (stream of 1's and 0's)
 - Assembly languages (low-level CPU instructions)
 - **High-level languages** (**high-level instructions**)
- Translation of high-level languages
 - Interpreter (translation for each instruction)
 - **Compiler** (**translation once for all code**)
 - Hybrid (combination of the above)
- Types of programming languages
 - Functional (e.g. Lisp)
 - **Structured** (e.g. Pascal, **C**, Ada)
 - Object-oriented (e.g. C++, Java, Python)

EECS10: Computational Methods in ECE, Lecture 3

(c) 2005 R. Doemer

3

History of C

- Evolved from BCPL and B
 - in the 60's and 70's
- Created in 1972 by Dennis Ritchie (Bell Labs)
 - first implementation on DEC PDP-11
 - added concept of *typing* (and other features)
 - development language of UNIX operating system
- “Traditional” C
 - 1978, “*The C Programming Language*”, by Brian W. Kernighan, Dennis M. Ritchie
 - ported to most platforms
- ANSI C
 - standardized in 1989 by ANSI and OSI
 - standard updated in 1999

EECS10: Computational Methods in ECE, Lecture 3

(c) 2005 R. Doemer

4

Introduction to C

- What is C?
 - Programming language
 - high-level
 - structured
 - compiled
 - Standard library
 - rich collection of existing functions
- Why C?
 - de-facto standard in software development
 - code is portable to many different platforms
 - supports structured and functional programming
 - easy transition to object-oriented programming
 - C++ / Java
 - freely available for most platforms

EECS10: Computational Methods in ECE, Lecture 3

(c) 2005 R. Doemer

5

Our first C Program

- Program example: `HelloWorld.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 */
```

EECS10: Computational Methods in ECE, Lecture 3

(c) 2005 R. Doemer

6

Our first C Program

- Program comments
 - start with `/*` and end with `*/`
 - are ignored by the compiler
 - should be used to
 - document the program code
 - structure the program code
 - enhance the readability
- `#include` preprocessor directive
 - inserts a header file into the code
- standard header file `<stdio.h>`
 - part of the C standard library
 - contains declarations of standard types and functions for data input and output (e.g. function `printf()`)

```

/* 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 */

```

EECS10: Computational Methods in ECE, Lecture 3

(c) 2005 R. Doemer

7

Our first C Program

- `int main(void)`
 - main function of the C program
 - the program execution starts (and ends) here
 - `main` must return an integer (`int`) value to the operating system at the end of its execution
 - return value of 0 indicates successful completion
 - return value greater than 0 usually indicates an error condition
- function body
 - block of code (definitions and statements)
 - starts with an opening brace (`{`)
 - ends with a closing brace (`}`)
- `printf()` function
 - formatted output (to stdout)
- `return` statement
 - ends a function and returns its argument as result

```

...
/* main function */
int main(void)
{
    printf("Hello World!\n");
    return 0;
}
/* EOF */

```

EECS10: Computational Methods in ECE, Lecture 3

(c) 2005 R. Doemer

8

Our first C Program

- Program compilation
 - compiler translates the code into an executable program
 - `gcc HelloWorld.c`
 - compiler reads file `HelloWorld.c` and creates file `a.out`
 - options may be specified to direct the compilation
 - `-o HelloWorld` specifies output file name
 - `-ansi -Wall` specifies ANSI code with all warnings
- Program execution
 - use the generated executable as command
 - `HelloWorld`
 - the operating system loads the program (loader), then executes its instructions (program execution), and finally resumes when the program has terminated

EECS10: Computational Methods in ECE, Lecture 3

(c) 2005 R. Doemer

9

Our first C Program

- Example session: `HelloWorld.c`

```

east% mkdir HelloWorld
east% cd HelloWorld
east% ls
east% vi HelloWorld.c
east% ls
HelloWorld.c
east% ls -l
-rw-r--r--  1 doemer  faculty    263 Sep 28 22:11 HelloWorld.c
east% gcc HelloWorld.c
east% ls -l
-rw-r--r--  1 doemer  faculty    263 Sep 28 22:11 HelloWorld.c
-rwxr-xr-x  1 doemer  faculty   6352 Sep 28 22:12 a.out*
east% a.out
Hello World!
east% gcc -Wall -ansi HelloWorld.c -o HelloWorld
east% ls -l
-rwxr-xr-x  1 doemer  faculty   6356 Sep 28 22:17 HelloWorld*
-rw-r--r--  1 doemer  faculty    263 Sep 28 22:17 HelloWorld.c
-rwxr-xr-x  1 doemer  faculty   6352 Sep 28 22:12 a.out*
east% HelloWorld
Hello World!

```

EECS10: Computational Methods in ECE, Lecture 3

(c) 2005 R. Doemer

10

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...